EXHIBIT D

UNREDACTED VERSION OF DOCUMENT SOUGHT TO BE SEALED

IN THE UNITED STATES DISTRICT COURT
FOR THE NORTHERN DISTRICT OF CALIFORNIA
SAN FRANCISCO DIVISION

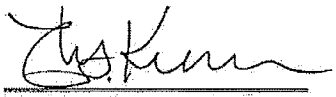
ORACLE AMERICA, INC.,	)	
	)	
Plaintiff,	)	
	)	
v.	)	Civil Action No. 10-03561 WHA
	)	
GOOGLE INC.,	)	
	)	
Defendant.	)	

EXPERT REPORT OF CHRIS F. KEMERER , Ph.D. REGARDING FAIR USE AND REBUTTAL TO GOOGLE'S OPENING EXPERT REPORTS

February 8, 2016

CONFIDENTIAL – ATTORNEYS' EYES ONLY
PURSUANT TO PROTECTIVE ORDER

I declare under penalty of perjury under the laws of the United States of America that the foregoing is true and correct. Executed this 8th day of February, 2016, in Pittsburgh, PA.



Chris F. Kemerer, PhD
Chris F. Kemerer PhD LLC

TABLE OF CONTENTS

I. Assignment

1. I have been asked to provide an overview and opinion from a technical perspective of fair use factor 3 as it arises in this case, and whether that factor, taken in its entirety, points towards a finding of fair use or not. I have been asked to provide opinions regarding the quantitative and qualitative amount and substantiality of the declaring code and structure, sequence and organization of the 37 specific Java Application Programming Interface packages ("Java APIs") copied by Google. I have been asked to respond to certain opinions of Dr. Owen Astrachan, Dr. Roderic Cattell and Mr. Andrew Hall, concerning factor 3 and other considerations asserted regarding fair use, including the quantitative and qualitative amount and substantiality of the Java APIs copied by Google, the value and the nature of the Java APIs, business models regarding control of software in general, and APIs in particular, the industry and Java ecosystem expectations regarding such control of APIs and software, and the economic consequences of such control. I have been asked to respond to certain opinions of Dr. Astrachan and Mr. Hall concerning factor 4 regarding the risk and uncertainty of GPL licensing options to Google.

2. In addressing these questions, I draw on my academic and professional background, which includes more than 30 years of experience in the areas of software engineering, technology adoption and diffusion, the creation and management of information systems, and the market economics of software and information systems.

3. I am being compensated for my work on this case at a rate of $595 per hour. My compensation is not contingent upon my testimony or on the result of this proceeding. Appendix B lists the materials I considered in preparing this report.

4. My work is ongoing, and I reserve the right to modify or supplement my conclusions as additional information becomes available to me, or as I perform further analysis.

II. Qualifications

5. I hold a Doctor of Philosophy (PhD) degree in (Information) Systems Sciences from the Graduate School of Industrial Administration at Carnegie Mellon University. I also hold a Master's Degree from Carnegie Mellon as well as a Bachelor's Degree in Decision Sciences and Economics from the Wharton School at the University of Pennsylvania.

6. Prior to my graduate studies I managed commercial software development projects in industry for a variety of public and private sector clients. I have also conducted and supervised a significant number of research projects involving the creation or measurement of software. Further, in my software development projects and my research, I have assessed technical and commercial risk regarding use of particular technologies and particular licensing terms.

7. I have been a full-time university professor for nearly 30 years conducting peer-reviewed research and teaching at the graduate level. I have held the David M. Roderick Chair of Information Systems at the University of Pittsburgh's Katz Graduate School of Business from 1995 to the present day. I also hold an adjunct professorship at Carnegie Mellon University, where I lecture on software engineering. I was previously a faculty member at the Massachusetts Institute of Technology's Sloan School of Management.

8. I have authored or co-authored more than 70 published articles in scholarly journals and am the editor of two books, one on software project management and one on information systems and industrial competitiveness. This research is consistently highly cited in both the software engineering and the business information systems literature. I have also authored two Harvard Business School-published case studies, including one on the mobile operating systems market.

9. My research has been funded by a variety of organizations, including the National Science Foundation. I have served in senior editorial roles in most of the leading journals in my field, including as Editor-in-Chief of *Information Systems Research*, Departmental Editor for Information Systems at *Management Science*, Senior Editor at *MIS Quarterly*, and Associate Editor at *IEEE Transactions on Software Engineering*. I have been honored to be named as a Distinguished Fellow of the INFORMS Information Systems Society.

10. Over the last two decades I have been retained as an outside independent expert witness over two dozen times in a variety of computer software-related matters, including software project management, anti-trust, and intellectual property issues.

III. Legal Principles: Copyright and Fair Use

11. It is my understanding that Copyright law is: "a form of protection provided . . . to the authors of 'original works of authorship,' including literary, dramatic, musical, artistic, and certain other intellectual property."[1] Copyright is also specifically endorsed in the U.S. Constitution: "The Congress shall have Power . . . To promote the Progress of Science and useful Arts, by securing for limited Times to Authors and Inventors the exclusive Right to their respective Writings and Discoveries."[2]

12. It is my understanding that certain copying activities may be considered "fair use" under the Copyright law, and if those activities are found to be fair use, then such copying may be permitted, as long the underlying purpose of copyright protection is not devalued, incentives to create are not thwarted, and the use does not injure the value of the copyrighted work. In order for unauthorized use of copyright-protected material to

[1] 17 U.S.C. § 106
[2] U.S. Const., Art. I, § 8

qualify as fair use, I understand that courts generally consider four factors: (1) the purpose and character of the use, (2) the nature of the copyrighted work, (3) the amount and substantiality of the portion taken, and (4) the effect of the use upon the potential market.[3]

IV. Summary of Opinions

13. Google's copying of thousands of lines of code and the structure, sequence and organization ("SSO") of the 37 Java APIs constituted the taking of a quantitatively and qualitatively large and substantial amount of valuable, copyright-protected material. Given this copying by Google, I conclude that fair use factor 3 ("the amount and substantiality" of the portion of the work used) weighs against a finding of fair use. I further respond to Dr. Astrachan's assertions that Google's taking was not quantitatively or qualitatively large or substantial. I disagree with Dr. Astrachan's assertions and it is my opinion that Dr. Astrachan undercounts and undervalues the code and SSO at issue. Dr. Astrachan's assertions do not establish that factor 3 weighs in favor of a finding of fair use.

14. Control of APIs, including particularly the Java APIs at issue, as well as other types of software development tools and programs, is a known and expected dynamic in the dissemination of software, and encourages innovation and investment by companies like Sun and Oracle to create new software. Given the long history, nature and widespread understanding of the practical and legal control exerted over the Java APIs, Google, developers and the Java industry and ecosystem did not actually or reasonably expect that the 37 Java APIs at issue could be used without permission or without constraints imposed by Sun and Oracle. Given these facts, I conclude that the assertions made throughout the reports of Dr. Astrachan and Dr. Cattell that the industry and developers purportedly expected that APIs in general, and the Java APIs in particular, could be copied and used without limitation are incorrect. To the contrary, it was well understood that APIs may be subject to restrictions on their copying and use, and the Java APIs were highly controlled in this regard. In my opinion, these facts weigh against a finding of fair use.

15. Google's speculative assertions regarding OpenJDK and any potential future use of code from OpenJDK under the GPLv2 with Classpath Exception license are inconsequential as they do not reflect the actual historic behavior at issue. In addition, based on that actual historic behavior, it is my opinion that Google and its customers and partners would not have accepted the significant business and technical risks of that license, and Google's customers and partners are not likely to accept such risks going forward. Given these facts, I conclude that fair use factor 4 (the effect of the use upon the potential market) weighs against a finding of fair use.

[3] 17 U.S.C. § 107

V. Structure of this Report

16. My opinions in this report are organized as follows:

17. In §VI, I consider the quantitative and qualitative amount and substantiality of the copied 37 Java APIs in relation to the Java SE platform. In particular, I conduct an analysis of how *central* the 37 Java APIs are to the Java SE platform by using a metric called *PageRank*. Using PageRank, I assess the importance of the Java APIs that Google copied to the Java platform, as compared to the Java APIs that Google did not copy. I find that the 37 Java SE APIs are highly central within Java SE compared to the non-copied APIs and, thus, Google copied a substantial portion of the Java SE platform.

18. In §VI-A, I conduct alternative PageRank analyses of the 37 Java APIs to the Java platform, accounting for, and in rebuttal to, arguments made by Dr. Astrachan. In particular, Dr. Astrachan asserts at one point in his report that 61 particular classes are necessary to use the Java programming language and asserts at another point in his report that the package java.lang is necessary to use the Java programming language. Without agreeing with his conclusion, but for purposes of completeness, and to rebut these arguments, I conduct sensitivity analyses that carry out PageRank analyses in a manner that excludes the material that Dr. Astrachan asserts is necessary to the Java programming language.[4] In my first sensitivity analysis, I exclude the 61 particular classes cited by Dr. Astrachan and then assess the importance of the remaining classes within the 37 Java APIs, using the PageRank metric. In my second sensitivity analysis, I exclude the package java.lang cited by Dr. Astrachan and then assess the importance of the remaining classes within the 37 Java APIs, using the PageRank metric. I refer to these analyses as sensitivity analyses, because they determine how sensitive the PageRank results are to exclusion of the 61 classes or the java.lang package. The results of these analyses are that even when I exclude the 61 classes or the java.lang package, I find that the remaining portion of the 37 Java APIs is still highly central within Java SE compared to the non-copied APIs and, thus, even under Dr. Astrachan's position, Google copied a substantial portion of the Java SE platform.

19. In §VI-B, I consider the quantitative and qualitative amount and substantiality of the copied 37 Java APIs in relation to the Android platform. While it is my understanding that factor 3 only appropriately considers "the amount and substantiality of the portion used in relation to the copyrighted work as a whole" rather than the portion used in relation to the infringing work, because Dr. Astrachan compares the 37 Java APIs to the Java platform, I respond to those assertions, through three modes of analysis, as follows.

[4] For the same reasons, as discussed further below, I conduct sensitivity analyses of the importance and centrality of the 37 Java APIs to Android, in a manner that excludes the material that Dr. Astrachan asserts is necessary to the Java programming language. Without agreeing with Dr. Astrachan's conclusions, I conduct these sensitivity analyses solely for the purposes of completeness and to demonstrate that, even excluding the material addressed under Dr. Astrachan's analysis, the 37 Java APIs are a substantial portion of the Java platform and a substantial taking by Google.

20. First, in §VI-B-1a, I consider the importance of the copied 37 Java APIs in relation to the Android platform by assessing whether the Android platform will compile if the declaring code of the 37 Java APIs is removed from Android. If the declaring code of the 37 Java APIs is removed from Android, Android will not compile. From this I conclude that the 37 Java APIs are important to the Android platform, and thus Google copied a substantial portion of the Java SE platform. In §VI-B-1b, I conduct an alternative compilation analysis of the 37 Java APIs, accounting for, and in rebuttal to, arguments made by Dr. Astrachan that either 61 classes or the java.lang package are necessary to use the Java programming language. In my alternative analysis I assess whether the Android platform will compile if the declaring code of the 61 classes or the java.lang package are left in Android, but the remaining declaring code of the 37 Java APIs is removed from Android. Under those conditions, Android still will not compile. From this I conclude that the 37 Java APIs are important to the Android platform, even under Dr. Astrachan's position and, thus, Google copied a substantial portion of the Java SE platform.

21. Second, in §VI-B-2a, I conduct an analysis of how central the 37 Java APIs are to the Android platform by using the PageRank metric. Using PageRank, I assess the importance of the copied Java APIs to the Android platform, as compared to the Java APIs that Google did not copy. I find that the 37 Java APIs are highly central within the Android platform compared to the non-copied APIs and, thus, Google copied a substantial portion of the Java SE platform. In §VI-B-2b, I conduct alternative PageRank analyses of the 37 Java APIs, accounting for and in rebuttal to arguments made by Dr. Astrachan that 61 classes or the java.lang package are necessary to use the Java programming language. I refer to these analyses as sensitivity analyses. Even when I exclude the 61 classes or the java.lang package, I find that the remaining portion of the 37 Java APIs are highly central within the Android platform compared to the non-copied APIs and, thus, even under Dr. Astrachan's position, Google copied a substantial portion of the Java SE platform.

22. Third, in §VI-B-3, I conduct an analysis of the importance of the 37 Java APIs to the Android platform by assessing the degree to which they contributed to the stability of the Android platform. In this analysis I consider the relative change behavior of the 37 Java APIs copied in Android, as compared to other APIs in the Android core libraries and frameworks. There is much less change and thus much greater stability of the 37 Java APIs, compared to other APIs in Android. This suggests that the stability of Android is driven by the 37 Java APIs, and that this material is important to Android. Thus, Google copied a substantial portion of the Java SE platform. I conduct alternative stability analyses of the 37 Java APIs to the Android platform, accounting for, and in rebuttal to, arguments made by Dr. Astrachan that 61 classes or the java.lang package are necessary to use the Java programming language. I refer to these analyses as sensitivity analyses. Even when I exclude the 61 classes or the java.lang package, I find that the remaining portion of the 37 Java APIs exhibited much less change than the other APIs in Android, suggesting that the stability of Android is driven

by these portions of the 37 Java APIs, and therefore they are important to Android. Thus, even under Dr. Astrachan's position, Google copied a substantial portion of Java SE platform.

23. In Section §VI-C, I consider the number of lines of code copied and assess the significance of the volume of code copied. Google copied approximately 11,000 lines of code. Dr. Astrachan states that Google copied 7,000 lines of code. I conclude that objectively, even if under-counted as 7,000 lines, this is still a large amount of copied code. It is also a substantial taking because those lines reflect structure, sequence and organization, and they are central and important to the Java platform. This is reflected both by Google's own statements about the importance of this code and by reference to other important software systems that are comprised of approximately this volume of code. Further, in §VI-C, I respond to Dr. Astrachan's analysis of the amount of copied code. I conclude that Dr. Astrachan inappropriately reduces the lines of code and the structure, sequence and organization to individual elements, in order to artificially undervalue both the amount and substantiality. I also conclude that Dr. Astrachan inappropriately compares the number of lines of code to Android, which is not an appropriate measure of amount and substantiality pursuant to the Court of Appeals opinion. I further conclude that Dr. Astrachan inappropriately compares the volume of copied code to the entire volume of code in the Java platform, because the declaring code is what is known to the audience of developers whereas implementing code is subordinate to this declaring code. And, even if a simple percentage calculation were to be done, a more pertinent comparison would be to put the copied lines of code in context by comparing the number of copied lines of declaring code to the total number of lines of declaring code in the 37 API packages in the Java SE platform. As a percentage, this more pertinent measure of the copied code is very high.

24. In §VII, I respond to other considerations that Dr. Astrachan and Dr. Cattell put forward, generally, as purportedly pertinent to fair use. In particular, Dr. Astrachan and Dr. Cattell repeatedly argue in general throughout their reports, and independent of any particular fair use factor, that there was a broad expectation across the industry and among developers that all APIs, including the 37 Java APIs at issue, could be freely copied and used, without limitation, restriction or control. Dr. Astrachan and Dr. Cattell argue that this assertion is purportedly material to fair use. I do not agree with this factual analysis or the conclusion. In this discussion I establish that control over, and restrictions on, the copying and use of software generally, and APIs and similar development tools in particular, has long been an expected and understood reality in the software industry. I discuss that such control and restrictions on copying and use of software generally, and APIs in particular, create incentives to invest in creation of such software, and therefore furthers the economic policies that motivate copyright law. In these sections I provide historical examples of control over APIs, evidence that the software industry has long understood that APIs, and the Java APIs in particular, are subject to restrictions on copying and use. I provide evidence that control over copying and use of APIs is expected in the burgeoning

"API economy" and evidence that such control incentivizes creation in this economy. I provide evidence that Google has always placed numerous limitations on copying and use of its own APIs. These facts weigh against a finding of fair use and rebut Dr. Cattell's and Dr. Astrachan's assertions regarding industry or developer expectations.

25. Finally, in §VIII, I rebut Dr. Astrachan's assertion that the mere availability of OpenJDK, a version of Java licensed under the GPLv2 with Classpath Exception license and any potential use of code from OpenJDK, is purportedly pertinent to fair use factor 4 (the impact that Google's use of the 37 Java APIs had on the potential or actual market for Java). I disagree with Dr. Astrachan's assertion because there is extensive historical evidence that Google and its customers and partners would not accept the GPLv2 with Classpath Exception license and, thus, would not use the OpenJDK code. Dr. Astrachan's assertion is speculative as it does not reflect the actual historic behavior at issue. I further disagree with Dr. Astrachan's assertion because there has been, and continues to be, considerable business and technical risk related to use of code made available under GPLv2 with Classpath Exception. For these reasons it is my opinion that OpenJDK is not pertinent to the impact of Google's use on the market for Java, because Google would never have accepted the license for OpenJDK. I conclude therefore that factor 4 weighs against a finding of fair use.

VI. Factor 3: Google's Copying was Quantitatively and Qualitatively Extensive and Substantial

26. I understand that fair use factor 3 considers "the amount and substantiality of the portion used in relation to the copyrighted work as a whole."[5] I further understand that analysis of this factor is viewed in the context of the copyrighted work and that "a taking may not be excused merely because it is insubstantial with respect to the infringing work."[6] I further understand that "the fact that a substantial portion of the infringing work was copied verbatim is evidence of the qualitative value of the copied material, both to the originator and to the plagiarist who seeks to profit from marketing someone else's copyrighted expression."[7]

27. Google's copying of thousands of lines of code and the structure, sequence and organization of the Java APIs constituted the taking of a quantitatively and qualitatively large and substantial amount of valuable, copyright-protected material. And these copied 37 Java APIs, even if considered only a small percentage of Java's and/or Android's overall code structure, represent its critical and valuable components. It is my opinion that these facts weigh against a finding of fair use.

[5] 17 U.S.C. § 107(3); United States Court of Appeals for the Federal Circuit, Oracle America Inc. v. Google Inc. 2013-1021, 1022, p. 58-59.
[6] United States Court of Appeals for the Federal Circuit, Oracle America Inc. v. Google Inc. 2013-1021, 1022, p. 58-59.
[7] United States Court of Appeals for the Federal Circuit, Oracle America Inc. v. Google Inc. 2013-1021, 1022, p. 58-59.

A. The Lines of Declaring Code and the Structure, Sequence and Organization of the 37 API Packages Copied by Google are Central to the Java Platform

28. *Centrality* is a metric that is used to describe the relative importance of a particular entity, or node, within a network of interconnected entities.[8] A connection between any two nodes can be described as a dependency. By treating all of the classes in the Java source code as discrete nodes, and all of the connections between classes as dependencies, the entirety of the Java source code can be analyzed as a network, and an impression of the overall contours of influence and interactivity within that network emerges.

29. The centrality of an individual node is roughly proportional to the number of connections that a particular node has to other nodes. An individual node with multiple connections to other nodes will have a higher centrality score than a node with only one connection.

30. Applying the notion of network centrality to the 37 Java API packages within the context of the Java SE 5 source code as a whole is a useful way to understand how important those packages are to Java.

31. A high centrality score for the 37 Java API packages would indicate that the classes inside them are connected to a high number of classes in packages outside those packages. Such a pattern would indicate that the rest of the Java source code depends heavily on the 37 Java APIs.

32. There are a variety of approaches to calculating the centrality of a node, and these measures differ in the balance of inputs that they consider with respect to the quantitative and qualitative nature of nodes and connections. A widely recognized metric is Google's PageRank.[9]

33. PageRank is a centrality measure that was developed by Larry Page and Sergey Brin as part of their research to develop a new search engine, and the mechanics of the PageRank algorithm were published by Page in 1998.[10] It was later implemented in the Google search engine, and is mentioned in the first patents filed describing its search methods.[11]

34. Generally, an entity's PageRank score is proportional to the number of connections that it has, with each connection in turn being weighted by the PageRank of the entity it connects to. This means that calculation of PageRank is a recursive and mathematically cumbersome task. Ultimately, the PageRank scores of all entities

---

[8] *See* Introduction to Social Network Methods, R. A. Hanneman, http://faculty.ucr.edu/~hanneman/nettext/C10_Centrality.html (accessed Feb. 8, 2016) for a discussion on the notion and implications of centrality using a social network as an analogy.
[9] *See* PageRank Centrality, http://www.sci.unich.it/~francesc/teaching/network/pagerank (accessed Feb. 8, 2016) for a discussion on the motivation and advantages of PageRank's use in calculating network centrality.
[10] *See* The PageRank Citation Ranking:
Bringing Order to the Web, L. Page, http://ilpubs.stanford.edu:8090/422/1/1999-66.pdf (accessed Feb. 8, 2016)
[11] *See* Method for Node Ranking in a Linked Database, US Patent No. 6285999 B1, http://www.google.com/patents/US6285999.

within a particular network form a probability distribution that totals one. For this reason, an entity's PageRank can be thought of as the probability that a random walk within the network will lead to that specific node.[12]

35. PageRank has since become a widely referenced tool for network analysis, and has been rigorously implemented in fields outside of web search[13] and has specifically been used to understand networks of Java classes.[14] For these reasons, it is an appropriate metric for evaluating the centrality of the 37 Java API packages in the Java SE 5 source code.

1) PageRank Results for the Java Platform

36. Examining the Java SE 5 platform as an interconnected network involves treating every class in the source code as a node in the greater network. Based on this approach a software tool called Understand is used to analyze intra-class dependencies and discern the broader network structure by building upwards from individual node dependencies.[15]

37. The Java software system network used in this analysis considers every public Java API class from the Java SE 5 source code.[16] The analysis aims to identify the extent to which the Java SE 5 source code leverages the functionality provided by the 37 Java API packages. Once created, this network can then be analyzed using a software tool called NetworkX, originally developed by the Los Alamos National Laboratory.[17] NetworkX provides tools for characterizing networks through computational means, including centrality scores.

38. NetworkX was used to analyze the centrality of the Java SE 5 software system by computing a PageRank score for each class inside the network. The results were then analyzed to identify the scores for only those

---

[12] *See* L. Page,, Section 2.5, for a discussion of the 'Random Surfer Model' that serves as the basis for this interpretation of a PageRank score.

[13] *See* "Identifying Critical Attack Assets in Dependency Attack Graphs," R.Sawilla and X. Ou, http://people.cis.ksu.edu/~xou/publications/drdc08.pdf (accessed Feb. 8, 2016), used here to assess vulnerabilities in security networks; *also see* "Network-Based Analysis of Software Change Propagation," R. Wang, R. Huang and B. Qu, http://www.hindawi.com/journals/tswj/2014/237243/ (accessed Feb. 8, 2016) (used here to understand change propagation in object-oriented software systems).

[14] Puppin, D. and F. Silvestri, "The Social Network of Java Classes," in *Proceedings of the 2006 ACM Symposium on Applied Computing*, pp. 1409-1413, ACM, 2006, *available at* http://pomino.isti.cnr.it/~silvestr/wp-content/uploads/2011/02/sac2006.pdf; OAGOOGLE0000609523; OAGOOGLE0007356223

[15] *See* SciTools Understand, https://scitools.com/features/#feature-category-dependency-analysis for a detailed discussion of the code dependency analysis features offered by Understand.

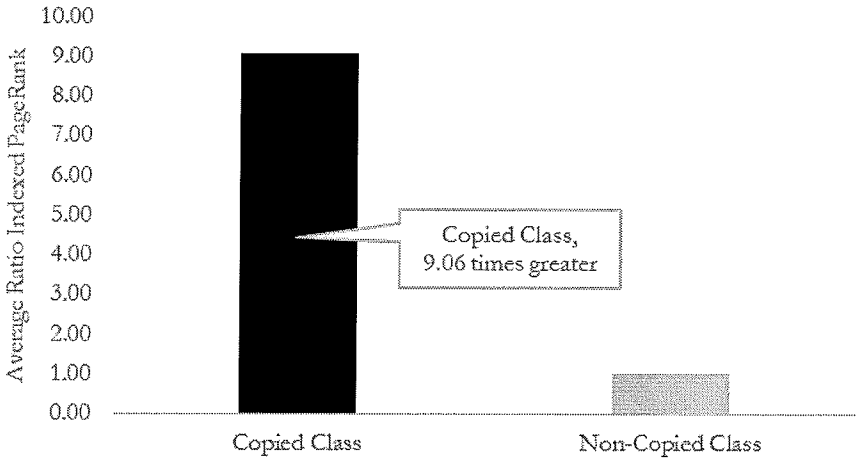
[16] TX 623 (Java SE 5 source code) Java SE 5 source code available at "Java SE 5.0 Downloads," Oracle, http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-javase5-419410.html (accessed Feb. 8, 2016) For JDK Java APIs, only publicly facing APIs are included in this analysis. This excludes the following 5 APIs from use in this analysis: javax.security.cert, javax.crypto, javax.crypto.interfaces, javax.crypto.spec, javax.net, and javax.net.ssl.

[17] *See* High-Productivity Software for Complex Networks, NetworkX, http://networkx.github.io/.

classes which belong to the 37 Java API packages. The classes in the 37 Java API packages could then be compared to the rest of the classes across the wider Java SE 5 source code (i.e. the non-copied classes).[18]

39. The results of this analysis show, through a comparison of PageRank scores of copied and non-copied classes, the relative centrality of the code that Google copied to the Java platform. The results are reflected in data in Appendix C, and in Figure 1 through Figure 6, below. Appendix C shows the PageRank scores of the classes in the copied 37 Java API packages, including the scores of both the copied classes and non-copied classes. Figure 1 shows, in visual form, a comparison of average PageRank scores of copied classes and non-copied classes in Java SE 5, each of which is normalized by the average score of the non-copied classes.[19] This figure clearly illustrates the greater PageRank scores of copied classes, with the average copied class boasting a score that is over 9 times greater than that of the average non-copied class.[20] These PageRank results show the importance of the 37 copied Java API packages to the network of the Java SE 5 platform.[21]

Figure 1: Average PageRank score of a copied class to a non-copied class code in Java SE 5



40. From examining the above results, one might ask whether the very high average PageRank score of the classes in a copied package may arise from the contribution of a few outlier classes that have particularly high

[18] Note that two classes, java.lang.Override and java.lang.annotation.Inherited, do not have any of the dependency types considered in this analysis. Since these classes are not connected to any other nodes in the Java SE 5 network, NetworkX excludes these classes with no identified dependencies.

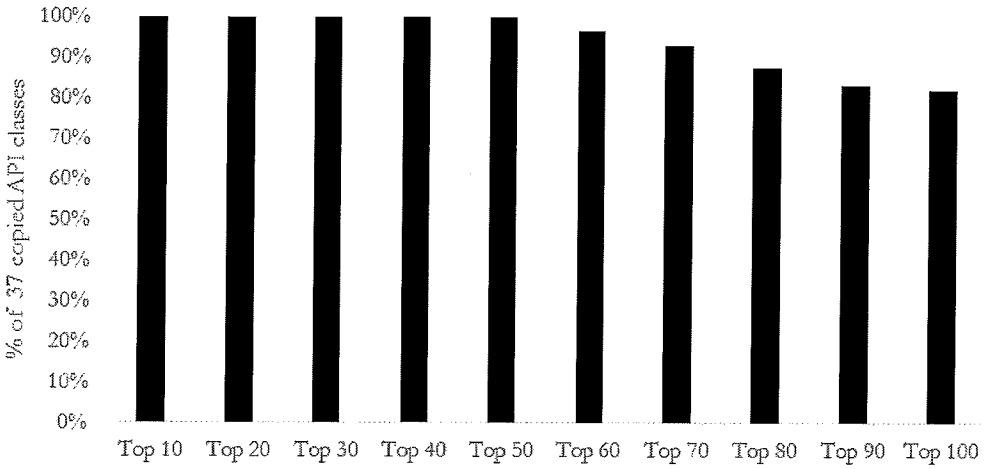
[19] To measure the relative importance of the copied classes to the non-copied classes, I bring the scores to the same scale by dividing both scores by the score of the non-copied packages. This normalization technique helps us clearly understand how large or small the score of the copied classes are compared to the scores of the non-copied classes.

[20] Average for the copied classes was 0.000361029 and for the non-copied classes was 4.19144E-05. PageRank scores for the copied classes appear in Appendix C.

[21] Recall that the PageRank calculation method demands that the sum of the PageRank score of every entity in the network must add up to 1. The Java SE 5 platform system network consists of over 6500 entities, all of whose PageRank scores must sum to 1. Due to its large size, the PageRank score of any one entity will be very small. This has no implications on the findings of the analysis - it is simply a result of using the PageRank measure on a network of this size.

PageRank scores. To address this, I calculated the percentage of the copied classes appearing within the top 10 ranked classes overall within Java SE 5. I then performed the same calculation for the top 20, top 30 and so on, until I had accounted for the top 100 ranked classes within Java SE 5. The results are shown in Figure 2, below. I find that nearly 100% of the top 10, 20, 30, 40, and 50 classes within Java SE 5 are classes that Google copied. Even when you look at the top 100 classes within Java SE 5, more than 80% of them are classes that Google copied. These findings, coupled with those discussed above, support the conclusion that the copied classes are central to the Java SE platform.

Figure 2: Percentage of classes with the highest PageRank score that belong to one of the 37 packages for various different rank group sizes



41. The leftmost column of Figure 2 shows that 100% of the 10 classes with the highest PageRank score within Java SE are classes that Google copied. The percentage of the top 20, 30, 40, 50 etc. classes within Java SE that Google copied is reflected in subsequent columns. The rightmost column shows that approximately 82% of the 100 classes with the highest PageRank scores within Java SE are classes that Google copied. As discussed above, from this figure it can be seen that whether one is considering only the top 10 ranked classes within Java SE or the top 100 ranked classes within Java SE, or any group of the highest ranked classes in between, the classes that Google copied feature prominently. This shows that the classes Google copied are of consistently high centrality to the Java SE platform.

2) Sensitivity Analysis: PageRank analysis and results under Google's assumptions

42. In Dr. Astrachan's report, at ¶164, he asserts that the 61 classes listed in Trial Exhibit 1062 are purportedly necessary to the Java programming language and cites related trial testimony. At ¶163, he asserts that the java.lang package is purportedly necessary to use the Java programming language. As discussed in Dr.

Schmidt's Rebuttal Report at ¶¶ 246, 247, 265-267, Dr. Astrachan's assertions in this regard are flawed. However, in the interest of completeness, and in response to Dr. Astrachan's arguments, I have conducted the PageRank analysis discussed above in a manner that excludes dependencies on the 61 classes and the java.lang package, respectively, in order to understand the centrality of the classes of the 37 Java APIs outside of the material that Dr. Astrachan asserts is necessary to the language. I refer to these analyses as sensitivity analyses, because they determine how sensitive the PageRank results are to exclusion of the 61 classes or the java.lang package in the ranking comparison. The results of the sensitivity analysis excluding the 61 classes in the ranking comparison are set forth immediately below, followed by the sensitivity analysis excluding the java.lang package in the ranking comparison. As demonstrated below, even when the dependencies on the 61 classes or the java.lang package are excluded, the remaining material in the 37 Java APIs have consistently high PageRank scores and are highly central to the Java SE platform.

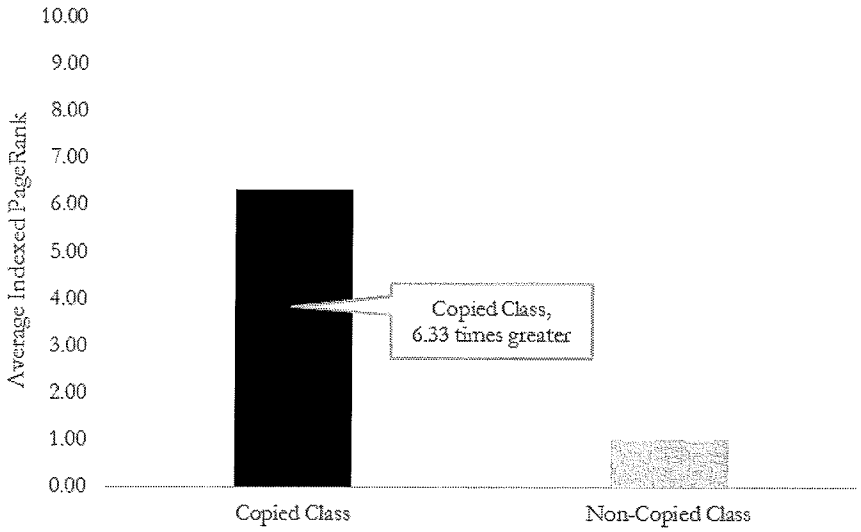
a) 61 Class Sensitivity Analysis

43. Dr. Astrachan asserts that the 61 classes listed in Trial Exhibit 1062 are purportedly necessary to use the Java programming language. While this analysis is flawed, as set forth in ¶¶ 244, 245, 264 of Dr. Schmidt's Rebuttal Report, in order to account for and rebut this argument I conduct the same PageRank analysis discussed above, but in a manner that excludes the 61 classes listed in Trial Exhibit 1062 from the output dataset of NetworkX.[22] The PageRank scores of the remaining classes in the 37 Java API packages were then compared to the rest of the class groupings across the wider Java SE 5 source code (i.e. the non-copied classes). Appendix C shows the average PageRank score of both the copied classes and non-copied classes (excluding the 61 classes from the count), each of which is normalized by the average score of the non-copied classes.

44. Figure 3, below, demonstrates the still greater PageRank scores of the copied classes even after excluding the 61 classes in the comparison, with the average remaining copied class boasting a score that is over 6 times greater than that of the average non-copied class. These PageRank results show that the 37 copied Java API packages are important to the network of the Java SE 5 platform, even accounting for Dr. Astrachan's argument regarding the 61 classes and removing those from the ranking comparison.

[22] Note that two classes, java.lang.Override and java.lang.annotation.Inherited, do not have any of the dependency types considered in this analysis. Since these classes are not connected to any other nodes in the Java SE 5 network, NetworkX eliminates these classes with no identified dependencies.

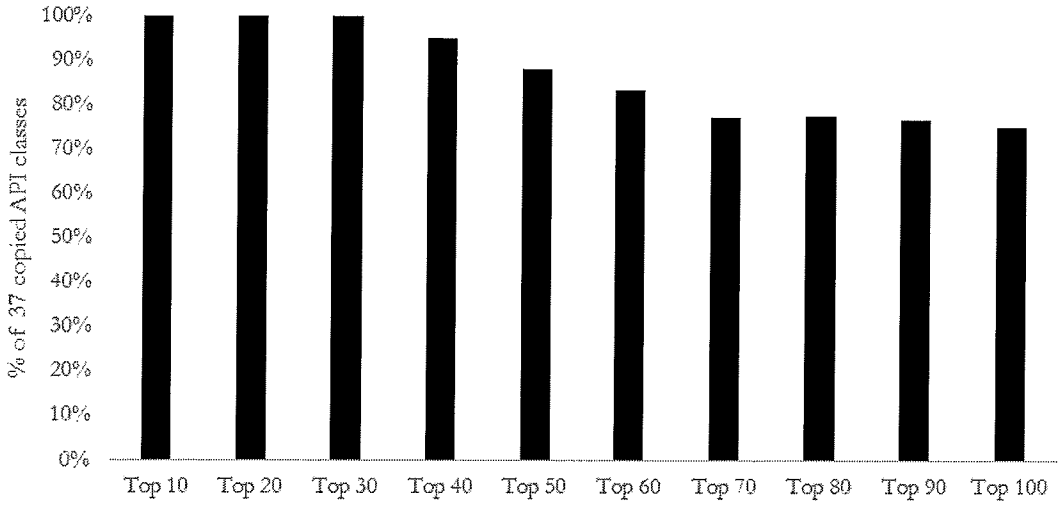
Figure 3: Average PageRank score of a copied class to a non-copied class code – 61 class scenario



45. Similar to the PageRank analysis set forth earlier in this report, I also calculated the percentage of the copied classes appearing within the top 10, 20, 30, 40, 50 etc. most highly ranked classes within the Java SE platform, up to the top 100 most highly ranked classes. However, this time, I excluded the 61 classes that Dr. Astrachan argues are necessary to the Java language. The results are shown in Figure 4, below. Even when you look at the top 100 most highly ranked classes, approximately 75% of them are classes that were copied by Google. Based on these results, the exclusion of the 61 classes in Dr. Astrachan's argument has no substantive effect on the conclusion drawn from the earlier PageRank calculations. Even excluding the 61 classes, the 37 copied API packages are still highly central to Java SE 5.[23]

[23] The decline in the PageRank score is to be expected, given that we are removing copied classes that contribute to the original score. These classes were excluded from the analysis only to accommodate Dr. Astrachan's alternative assertions. However, despite this, the ratio of scores remains highly skewed, with a PageRank score of 6.33 even in the reduced dataset. In addition, it is still the case that when looking even at only the Top 100 highest ranking classes, the remaining copied classes still account for the overwhelming portion of the indexed rank scores, supporting the notion that the copied classes are highly central to Java SE 5 even under Dr. Astrachan's alternative assertions.

Figure 4: Percentage of classes with the highest PageRank score that belong to one of the 37 packages for various different rank group sizes – 61 class scenario



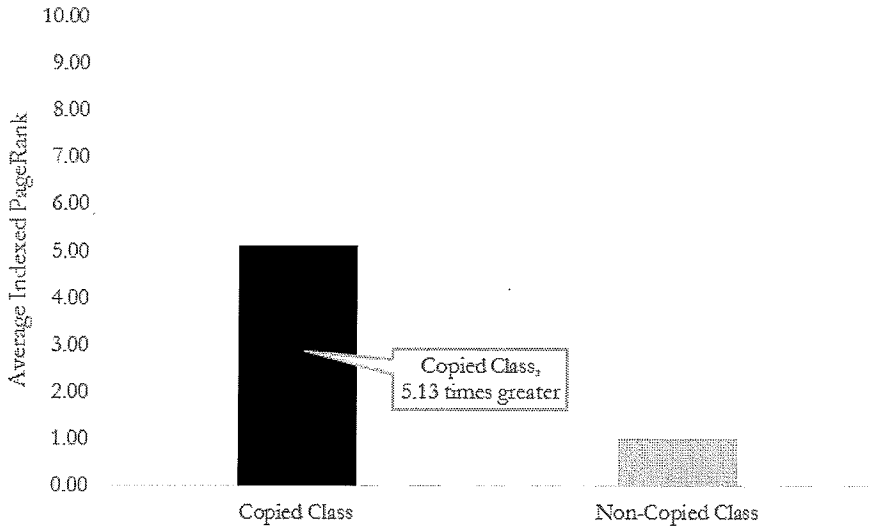
46. In summary, this sensitivity analysis demonstrates that the removal of these 61 classes in the comparison has no substantive effect on the PageRank results. This is because in the original PageRank results discussed earlier with all classes included, 82 of the classes ranked in the top 100 in the PageRank results are copied classes, and 152 of the classes ranked in the top 200 are copied classes. The ratio between the average PageRank score of copied classes and non-copied classes decreases from 9.06 to 6.33. But the average PageRank score of the copied classes is still greater than that of the non-copied classes. In the new ranking without the 61 classes, there are still 75 copied classes in the top 100 ranked classes in the PageRank results. As a result, the removal of the 61 classes has no substantive effect on the PageRank results, which, even excluding the 61 classes, show that the classes copied by Google are highly central to the Java SE platform relative to the non-copied classes.

b) Java.lang Package Sensitivity Analysis

47. Dr. Astrachan asserts that the java.lang package is purportedly necessary to use the Java programming language. While this analysis is flawed, as set forth in ¶¶ 244, 245 and 264 of Dr. Schmidt's Rebuttal Report, in order to account for and rebut this argument I conducted the same PageRank analysis discussed above, but with removing the java.lang package from the output dataset of NetworkX to produce PageRank scores of each entity in the Java SE 5 platform. The PageRank scores of the remaining classes in the 37 Java API packages were then compared to the rest of the class groupings across the wider Java SE 5 source code (i.e. the non-copied classes). Appendix C shows the average PageRank score of both the copied classes and non-copied classes (excluding the java.lang package), each of which is normalized by the average score of the non-copied classes.

48. Figure 5, below, demonstrates the still greater PageRank scores of the copied classes even after removing the java.lang package, with the average copied class boasting a score that is over 5 times greater than that of the average non-copied class. These PageRank results show that the 37 copied Java API packages are important to the network of the Java SE 5 platform, even accounting for Dr. Astrachan's argument regarding the java.lang package and removing that from the analysis.

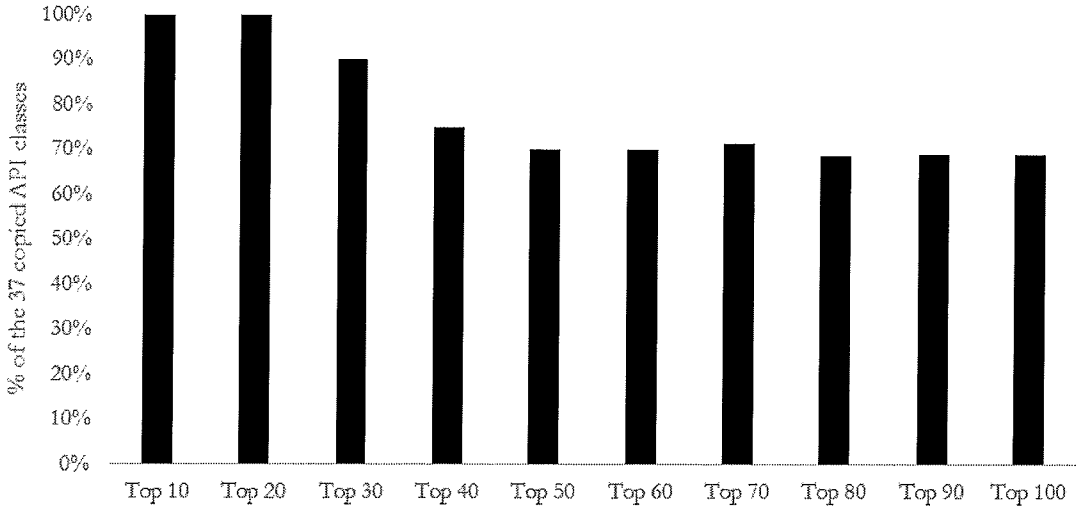
Figure 5: Average PageRank score of a copied class to a non-copied class code – java.lang package scenario



49. Similar to the PageRank analysis set forth earlier in this report, I also calculated the percentage of the copied classes appearing within the top 10, 20, 30, 40, 50 etc. most highly ranked classes within the Java SE platform, up to the top 100 most highly ranked classes within in the Java SE platform. However, this time, I excluded the java.lang package that Dr. Astrachan argues is necessary to the Java language. The results are shown in Figure 6, below. Even when you look at the top 100 most highly ranked classes, approximately 70% of them are classes that were copied by Google. Based on these results, the removal of the java.lang package in Dr. Astrachan's argument has no substantive effect on the conclusion drawn from the PageRank calculations. Even excluding the java.lang package, the 37 copied API packages are highly central to Java SE 5.[24]

[24] The decline in the PageRank score is to be expected, given that we are removing copied classes that contribute to the original score. These classes were excluded from the analysis only to accommodate Dr. Astrachan's alternative assertions. However, despite this, the ratio of scores remains highly skewed, with a PageRank score of 5.13 even in the reduced dataset. In addition, it is still the case that when looking even at only the Top 100 highest ranking classes, the remaining copied classes still account for the overwhelming portion of the indexed rank scores, supporting the notion that the copied classes are highly central to Java SE 5 even under Dr. Astrachan's alternative assertions.

Figure 6: Percentage of classes with the highest PageRank score that belong to one of the 37 packages for various different rank group sizes – java.lang package scenario



50. In summary, this sensitivity analysis demonstrates that the removal of the java.lang package has no substantive effect on the PageRank results. This is because in the original PageRank results detailed earlier with all classes included, 82 of the classes ranked in the top 100 in the PageRank results are copied classes, and 152 of the classes ranked in the top 200 are copied classes. This indicates that copied classes have a much higher average score than non-copied classes, and approximately 76% of the top 200 classes in the original results are copied classes. By removing the entire java.lang package, 41 classes of the top 100 classes in the original results are removed. The ratio between the average PageRank score of copied classes and non-copied classes does decrease from 9.06 to 5.13[25]. However, in the new ranking without the java.lang package, there are still 69 copied classes in the top 100 ranked classes in the PageRank results. As a result, the removal of the java.lang package has no substantive effect on the PageRank results, which, even excluding the java.lang package, show that the classes copied by Google are highly central to the Java SE platform compared to the non-copied classes.

3) Qualitative centrality of the Java APIs to Java SE 5

51. The centrality of the 37 Java APIs to the Java SE 5 platform can be seen in other ways as well. The 37 Java APIs are special purpose Java API packages which enable developers to more quickly and efficiently develop applications and provide prewritten functionality. As discussed in the Rebuttal Report of Dr. Schmidt at §VIII-B, upon which I rely, the 37 Java APIs are a fundamental portion of the Java SE platform, and provide developers important capabilities. For example, many other tools within the Java SE platform are built on top of the 37 Java APIs. From a developer standpoint, the depth and breadth of the 37 Java APIs allows developers

[25] Raw PageRank scores for each class in Java SE can be found in Appendix C.

to focus on creating their own application code, rather than having to recreate the code of these pre-written programs. Developers come to know and appreciate the declaring code and SSO of these packages. The 37 Java APIs represent and express to developers information about prewritten functionality that is more important and valuable to them in the software development process than other very general purpose development tools and APIs in the platform. For example, as noted by Dr. Schmidt at §VIII-B, developers would not have a motivation to use general purpose tools in the Java SE platform, such as the javac compiler, unless they were already attracted to the platform by more specific Java APIs, which are prewritten programs that make development easier.

B. The Lines of Declaring Code and the Structure, Sequence and Organization of the 37 API Packages Copied by Google are Important to the Android Platform

52. For purposes of factor 3, the Court of Appeals has criticized comparison of the copied material to the infringing work (here Android)[26] since the relationship between the copied material and the infringing work is in the control of the alleged infringer. Contrary to this guidance, Dr. Astrachan repeatedly attempts in his analysis of fair use factor 3 to compare the 37 API packages to the Android platform. While it is my opinion that this comparison is immaterial to the analysis of fair use factor 3, in order to provide a more comprehensive rebuttal argument I conduct my own analysis of the importance and centrality of the 37 API packages to the Android platform.

53. First, based on the work of Dr. Schmidt, I consider the importance of the 37 API packages to Android by considering whether the Android platform will compile when the declaring code of the 37 API packages is removed from Android. Then, based on the work of Dr. Schmidt, I assess whether the Android platform will compile when the declaring code for the 61 classes or the java.lang package that Dr. Astrachan asserts to be part of the Java language are retained in the Android platform, but the remaining code of the 37 API packages is removed. Under each of these scenarios, the removal of the declaring code results in the inability of the Android platform to compile. On this basis I conclude that the declaring code and structure, sequence and organization of the 37 Java API packages are important to the Android platform. This confirms that even under Dr. Astrachan's arguments the copying was qualitatively substantial because what Google copied was critical to Android.

54. Second, I conduct an analysis of how central the 37 Java APIs are to the Android platform by using the PageRank metric. Using PageRank, I assess the importance of the Java APIs that Google copied to the Android platform, as compared to the Java APIs that Google did not copy. I find that the 37 Java APIs are highly central within the Android platform compared to the non-copied APIs and, thus, Google copied a substantial

[26] United States Court of Appeals for the Federal Circuit, Oracle America Inc. v. Google Inc. 2013-1021, 1022, p. 58 See page 58.

portion of the Java SE platform. I conduct alternative PageRank analyses of the 37 Java APIs to the Android platform, accounting for, and in rebuttal to, arguments made by Dr. Astrachan that 61 classes or the java.lang package are necessary to use the Java programming language. I refer to these analyses as sensitivity analyses. Even when I exclude the 61 classes or the java.lang package I find that the remaining portion of the 37 Java APIs are highly central within the Android platform compared to the non-copied APIs. This confirms that even under Dr. Astrachan's arguments the copying was qualitatively substantial because what Google copied was so central to Android.

55. Third, I conduct an analysis of the importance of the 37 Java APIs to the Android platform by assessing the degree to which they contributed to the stability of the Android platform. In this analysis I consider the relative changes in the method declarations of the 37 Java APIs copied in Android, as compared to other APIs in the Android core libraries and frameworks. There is much less change and thus much greater stability of the 37 Java APIs, compared to other APIs in Android. This suggests that the stability of Android is driven by the 37 Java APIs, and that they are therefore important to Android. Thus, Google copied a substantial portion of the Java SE platform. I conduct alternative stability analyses of the 37 Java APIs to the Android platform, accounting for, and in rebuttal to, arguments made by Dr. Astrachan that 61 classes or the java.lang package are necessary to use the Java programming language. I refer to these analyses as sensitivity analyses. Even when I exclude the 61 classes or the java.lang package, I find that the remaining portion of the 37 Java APIs exhibited much less change than the other APIs in Android, suggesting that the stability of Android is driven by these portions of the 37 Java APIs, and that this material is therefore important to Android. This confirms that even under Dr. Astrachan's arguments the copying was qualitatively substantial because what Google copied was so important to Android's stability.

1) Compilation Analysis Shows The Importance Of The 37 Java APIs To Android

a) The Android Platform will not compile without the declaring code and associated SSO of all or any one of the 37 Java API Packages

56. I have conferred with Dr. Schmidt who carried out an analysis of, and generated a report about, whether the Android platform properly compiles without all or any one of the 37 Java API packages. I rely upon and incorporate by reference in its entirety and refer to the contents of Dr. Schmidt's expert report. As detailed in Dr. Schmidt's report, the Android platform will not compile in the absence of the 37 API packages, or any one of them, or when the declaring code of any or all of the 37 API packages is removed.[27]

[27] Schmidt Opening Report, ¶¶ 90-97

57. Google's technical expert witness, Dr. Astrachan, has previously confirmed that this is the case: "Q. What would happen if you ripped those lines out of Android? A. Well, for the purposes of the Android core libraries, those are part of it, so they need to be there for Android to work as it's been designed." (Astrachan Tr. 2212)

58. On this basis I conclude that the declaring code and structure, sequence and organization of the 37 Java API packages is important to the Android platform. This confirms that the copying was qualitatively substantial because what Google copied was critical to Android.

b) Under Google's assumptions, the Android Platform will not compile without the declaring code and associated SSO of all or any one of the 37 Java API Packages

59. I have conferred with Dr. Schmidt who carried out an analysis of, and generated a report on, whether the Android platform properly compiles without all or any one of the 37 Java API packages, or the declaring code of any or all of the 37 API packages, but where the declaring code of the 61 classes or java.lang packages that Dr. Astrachan argues are necessary to the Java language remain intact. I rely upon and incorporate by reference in its entirety and refer to the contents of Dr. Schmidt's Opening Report. When the declaring code from the 61 classes or the java.lang packages are left undisturbed, but the remainder of the declaring code from the 37 API packages is removed, the Android platform still will not compile. (Schmidt Fair Use and Rebuttal Report, ¶¶ 264-267) On this basis, I conclude that the declaring code and structure, sequence and organization of the 37 Java API packages is important to the Android platform. This confirms that the copying was qualitatively substantial because what Google copied was critical to Android.

2) <u>PageRank Analysis Shows The Centrality Of The 37 Java APIs To Android</u>

a) PageRank Results for the Android Platform

60. As already discussed in the context of the Java platform, the PageRank metric can be used to measure the centrality of the 37 Java APIs to the Java platform. The same methodology can be used to measure the centrality of the 37 Java APIs to the Android platform.

61. Examining the Android operating system as an interconnected network involves treating every class in the source code as a node in the greater network. Based on this approach, a software tool called Understand is used to analyze intra-class dependencies and discern the broader network structure by building upwards from individual node dependencies.[28]

[28] *See* SciTools Understand, https://scitools.com/features/#feature-category-dependency-analysis for a detailed discussion of the code dependency analysis features offered by Understand.

62. The Android software system network used in this analysis considers every single Java class from Version 5.1.0, release 1 (Lollipop) of the Android Open Source Project (AOSP) source code. The analysis aims to identify the extent to which the Android source code leverages the functionality provided by the 37 Java API packages. The functionality provided by these packages can only be directly implemented by other Java source material. These facts require that only Java material from the AOSP source code be considered in the network analysis. That is, because the analysis considers dependencies on Java packages, it is appropriate to look at Android source code written in Java. It is not possible for the 37 Java packages to directly depend on portions of Android that are not written in Java (such as native code or binaries).

63. Once created, this network can then be analyzed using a software tool called NetworkX,[29] as discussed earlier in this report. NetworkX provides tools for characterizing networks through computational means, including centrality scores.

64. NetworkX was used to analyze the centrality of the Android software system by computing various metrics, including PageRank, for each class inside the network. The results were then processed to identify the scores for only those classes which belong to the 37 Java API packages. The classes in the 37 Java API packages could then be compared to the rest of the class groupings across the wider Android source code (i.e. the non-copied classes).

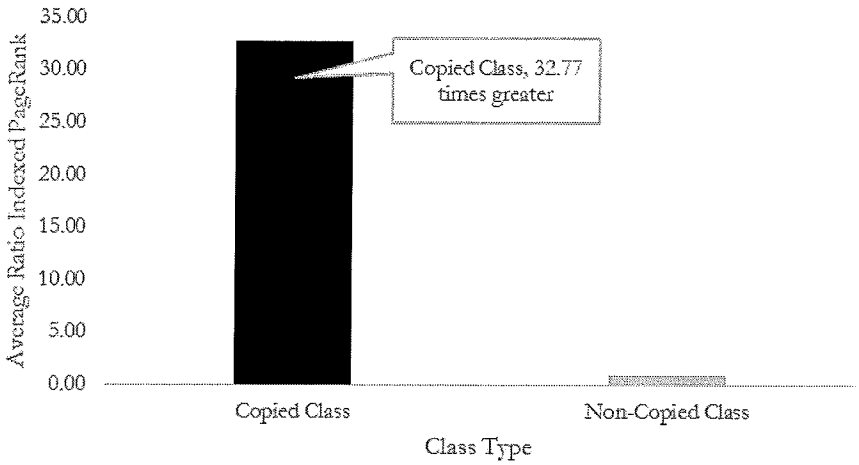
65. Appendix D shows the average PageRank score of both the copied classes and non-copied classes, each of which is normalized by the average score of the non-copied classes. Figure 7 shows below, in visual form, a comparison of average PageRank scores of copied classes and non-copied classes. This figure clearly demonstrates the greater PageRank scores of the copied classes with the average copied class boasting a score that is over 30 times greater than that of the average non-copied class. These PageRank results show the importance of the 37 copied Java API packages to the network of the Android operating system.[30]

66. The PageRank scores reported in Appendix D are normalized to capture the relative magnitudes of the PageRank scores of copied and non-copied classes. The actual magnitude of the raw scores themselves is not significant for the reasons stated above.

[29] *See* High-productivity software for complex networks," NetworkX, See http://networkx.github.io/.

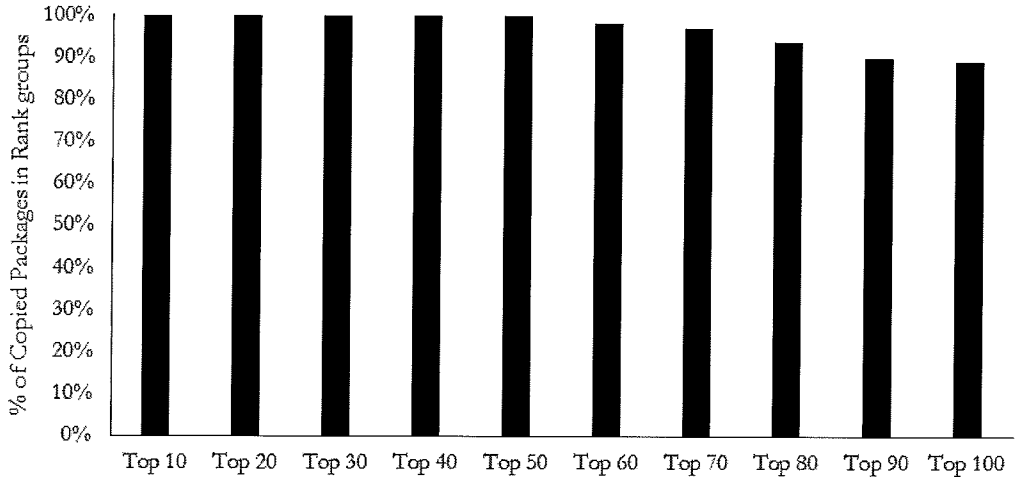
[30] Recall that the PageRank calculation method demands that the sum of the PageRank score of every entity in the network must add up to 1. The Android software system network consists of over 17,000 entities, all of whose PageRank scores must sum to 1. Due to its large size, the PageRank score of any one entity will be very small. This has no implications on the findings of the analysis - it is simply a result of using PageRank on a network of this size.

Figure 7: Average PageRank score of a copied class to a non-copied class code



67. From examining the above results one might ask whether the very high average PageRank score of the classes in a copied package may arise from the contribution of a few outlier classes that have particularly high PageRank scores. To address this, I calculated the percentage of the copied classes appearing within the top 10 ranked classes overall within Android. I then performed the same calculation for the top 20, top 30 and so on, until I had accounted for the top 100 ranked classes within Android. The results are shown in Figure 8, below. I find that nearly 100% of the top 10, 20, 30, 40, and 50 classes within Android are classes that Google copied. Even when you look at the top 100 classes within Android, nearly 90% of them are classes that Google copied. These findings, coupled with those discussed above, indicate to me that the copied classes are central to the Android platform.

Figure 8: Percentage of classes with the highest PageRank score that belong to one of the 37 packages for various different rank group sizes



68. The leftmost column of Figure 8 shows that 100% of the 10 classes with the highest PageRank score within Android are classes that Google copied. The percentage of the top 20, 30, 40, 50 etc. classes within Android that Google copied is reflected in subsequent columns. The rightmost column shows that nearly 90% of the 100 classes with the highest PageRank score within Android are classes that Google copied. As discussed above, from this figure it can be seen that whether one is considering only the top 10 ranked classes within Android or the top 100 ranked classes within Android, or any group of the highest ranked classes in between, the classes that Google copied feature prominently. This shows that the classes Google copied are of consistently high centrality to the Android platform.

b) Sensitivity Analysis: PageRank analysis and results under Google's assumptions

69. As discussed with regard to my PageRank analysis earlier in this report regarding the 37 Java APIs in the context of the Java SE platform, Dr. Astrachan asserts that 61 particular classes or the java.lang package are necessary to use the Java programming language.[31] As discussed in Dr. Schmidt's Rebuttal Report at ¶¶ 246, 247, 265-267, Dr. Astrachan's assertions in this regard are flawed. However, in the interest of completeness and in response to Dr. Astrachan's arguments, I have conducted the PageRank analysis discussed above in a manner that excludes dependencies on the 61 classes and the java.lang package, respectively, in order to understand the centrality of the classes of the 37 Java APIs outside of the material that Dr. Astrachan asserts is necessary to the language. I refer to these analyses as sensitivity analyses because they determine how sensitive the PageRank results are to exclusion of the 61 classes or the java.lang package in the ranking comparison. The

[31] Astrachan Opening Report, ¶¶ 163-164

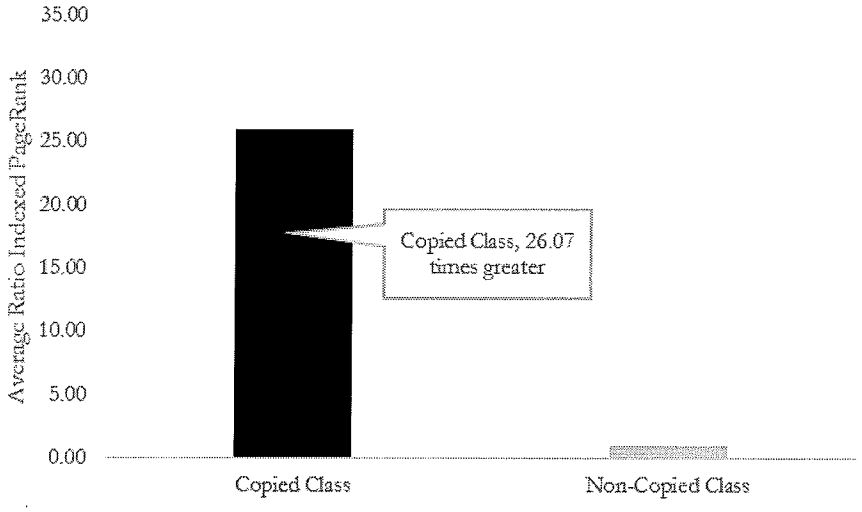
results of the sensitivity analysis excluding the 61 classes in the ranking comparison is set forth immediately below, followed by the sensitivity analysis excluding the java.lang package in the ranking comparison. As demonstrated below, even when the dependencies on the 61 classes or the java.lang package are excluded, the remaining material in the 37 Java APIs have consistently high PageRank scores, and are highly central to the Android platform.

(i) 61 Class Sensitivity Analysis

70. Dr. Astrachan asserts that the 61 classes listed in Trial Exhibit 1062 are purportedly necessary to use the Java programming language. (Astrachan Report, ¶ 164) While this analysis is flawed, as set forth in ¶¶ 46, 247 and 265-267 of Dr. Schmidt's Rebuttal Report, in order to account for and rebut this argument I conduct the same PageRank analysis discussed above, but in a manner that excludes the 61 classes from the output dataset of NetworkX. The PageRank scores of the remaining classes in the 37 Java API packages were then compared to the rest of the class groupings across the wider Android source code (i.e. the non-copied classes). Appendix D shows the average PageRank score of both the copied classes and non-copied classes (excluding the 61 classes from the count), each of which is normalized by the average score of the non-copied classes.

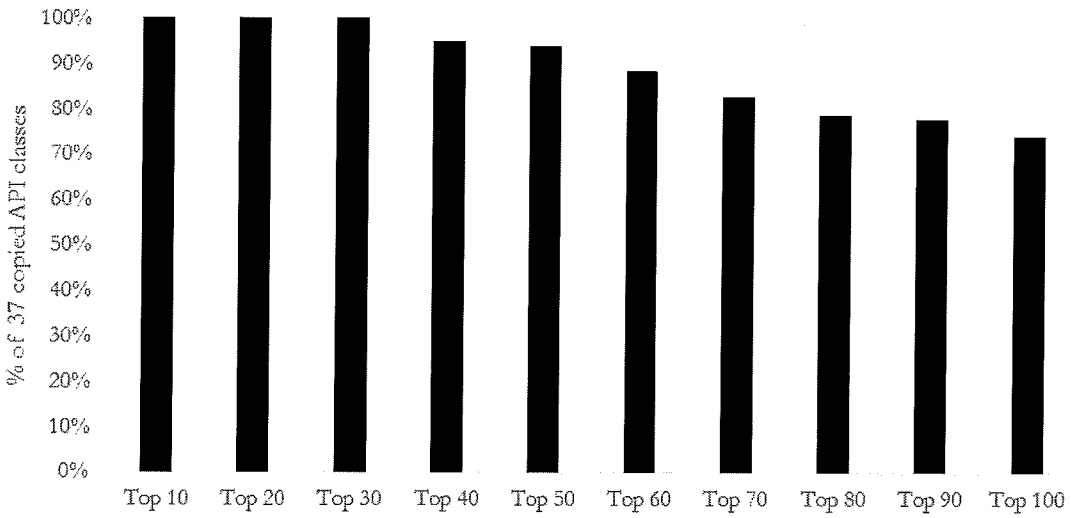
71. Figure 9, below, demonstrates the still greater PageRank scores of the copied classes even after excluding the 61 classes in the comparison, with the average remaining copied class boasting a score that is over 26 times greater than that of the average non-copied class. These PageRank results show that the 37 copied Java API packages are important to the network of the Android platform, even accounting for Dr. Astrachan's argument regarding the 61 classes and removing those from the ranking comparison.

Figure 9: Average PageRank score of a copied class to a non-copied class code – 61 class scenario



72. Similar to the PageRank analysis set forth earlier in this report, I also calculated the percentage of the copied classes appearing within the top 10, 20, 30 etc. most highly ranked classes within the Android platform, up to the top 100 most highly ranked classes. However, this time, I excluded the 61 classes that Dr. Astrachan argues are necessary to the Java language. The results are shown below in Figure 10. Even when you look at the top 100 most highly ranked classes, approximately 74% of them are classes that were copied by Google. Based on these results, the exclusion of the 61 classes in Dr. Astrachan's argument has no substantive effect on the conclusion drawn from the earlier PageRank calculations. Even excluding the 61 classes, the 37 copied API packages are still highly central to Android.[32]

Figure 10: Percentage of classes with the highest PageRank score that belong to one of the 37 packages for various different rank group sizes – 61 class scenario



73. In summary, this sensitivity analysis demonstrates that the removal of the 61 classes in the comparison has no substantive effect on the PageRank results. This is because in the original PageRank results shown earlier with all classes included, 89 of the classes ranked in the top 100 in the PageRank results are copied classes, and 148 of the classes ranked in the top 200 are copied classes. By removing the 61 classes, the ratio between the average PageRank score of copied classes and non-copied classes decreases from 32.77 to 26.07. But the average PageRank score of the copied classes is still greater than that of the non-copied classes. In the new ranking without the 61 classes, there are still 74 copied classes in the top 100 ranked classes in the PageRank results. As a result, the removal of the 61 classes has no substantive effect on the PageRank results, which,

[32] The decline in the PageRank score is to be expected, given that we are removing copied classes that contribute to the original score. These classes were excluded from the analysis only to accommodate Dr. Astrachan's alternative assertions. However, despite this, the ratio of scores remains highly skewed, with a PageRank score of 26.07 even in the reduced dataset. In addition, it is still the case that when looking even at only the Top 100 highest ranking classes, the remaining copied classes still account for the overwhelming portion of the indexed rank scores, supporting the notion that the copied classes are highly central to Android even under Dr. Astrachan's alternative assertions.

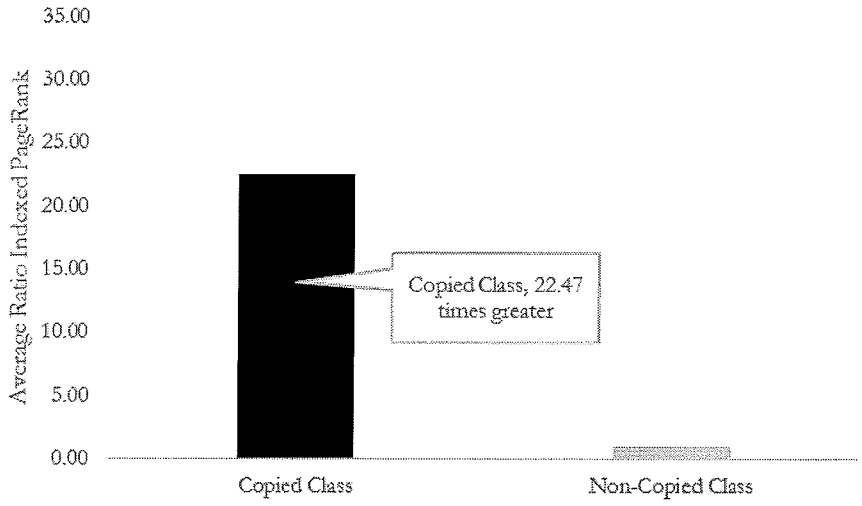
even excluding the 61 classes, show that the classes copied by Google are highly central to the Android platform, compared to the non-copied classes.

(ii) Java.lang Package Sensitivity Analysis

74. Dr. Astrachan asserts that the java.lang package is purportedly necessary to use the Java programming language. (Astrachan Report, ¶ 163) While this analysis is flawed, as set forth in ¶¶ 24, 245 and 264 of Dr. Schmidt's Rebuttal Report, in order to account for and rebut this argument I conducted the same PageRank analysis discussed above, but with removing the java.lang package from the output dataset of NetworkX to produce PageRank scores of each entity in the Android platform. The PageRank scores of the remaining classes in the 37 Java API packages were then compared to the rest of the class groupings across the wider Android source code (i.e. the non-copied classes). Appendix D shows the PageRank score of both the copied classes and non-copied classes (excluding the java.lang package), each of which is normalized by the average score of the non-copied classes.

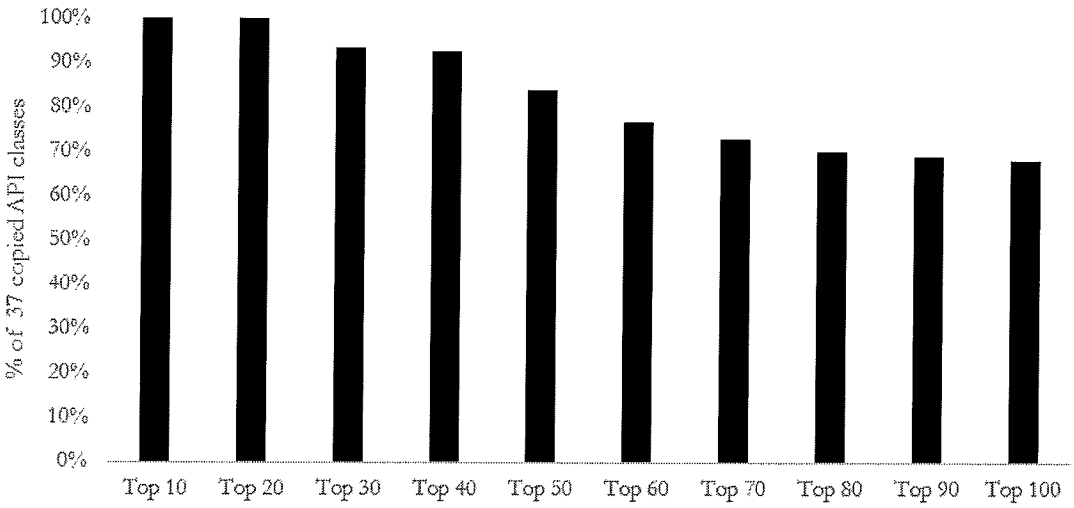
75. Figure 11, below, demonstrates the still greater PageRank scores of the copied classes even after removing the java.lang package, with the average copied class boasting a score that is over 22 times greater than that of the average non-copied class. These PageRank results show that the 37 copied Java API packages are important to the network of the Android platform, even accounting for Dr. Astrachan's argument regarding the java.lang package and removing that from the analysis.

Figure 11: Average PageRank score of a copied class to a non-copied class code – java.lang package scenario



76. Similar to the PageRank analysis set forth earlier in this report, I also calculated the percentage of the copied classes appearing within the top 10, 20, 30 etc. most highly ranked classes within the Android platform, up to the top 100 most highly ranked classes within in the Android platform. However, this time, I excluded the java.lang package that Dr. Astrachan argues is necessary to the Java language. The results are shown in Figure 12, below. Even when you look at the top 100 most highly ranked classes, approximately 68% of them are classes that were copied by Google. Based on these results, the removal of the java.lang package in Dr. Astrachan's argument has no substantive effect on the conclusion drawn from the PageRank calculations. Even excluding the java.lang package, the 37 copied API packages are highly central to the Android platform.[33]

Figure 12: Percentage of classes with the highest PageRank score that belong to one of the 37 packages for various different rank group sizes – java.lang package scenario



77. In summary, this sensitivity analysis demonstrates that the removal of the java.lang package has no substantive effect on the PageRank results. This is because in the original PageRank results discussed earlier with all classes included, 89 of the classes ranked in the top 100 in the PageRank results are copied classes, and 148 of the classes ranked in the top 200 are copied classes. This indicates that copied classes have a much higher average score than non-copied classes, and approximately 74% of the top 200 classes in the original results are copied classes. By removing the entire java.lang package, 42 classes of the top 100 classes in the original results are removed. The ratio between the average PageRank score of copied classes and non-copied

[33] The decline in the PageRank score is to be expected, given that we are removing copied classes that contribute to the original score. These classes were excluded from the analysis only to accommodate Dr. Astrachan's alternative assertions. However, despite this, the ratio of scores remains highly skewed, with a PageRank score of 22.47 even in the reduced dataset. In addition, it is still the case that when looking even at only the Top 100 highest ranking classes, the remaining copied classes still account for the overwhelming portion of the indexed rank scores, supporting the notion that the copied classes are highly central to Android even under Dr. Astrachan's alternative assertions.

classes decreases from 32.77 to 22.47.[34] However, in the new ranking without the java.lang package, there are still 68 copied classes in the top 100 ranked classes in the PageRank results. As a result, the removal of the java.lang package has no substantive effect on the PageRank results, which, even excluding the java.lang package, show that the classes copied by Google are highly central to the Android platform, compared to the non-copied classes.

3) Stability Analysis Shows The Importance Of The 37 Java APIs To Android

78. Another way to understand the importance of the 37 Java APIs to the Android platform is to consider the degree to which they contributed to the stability of the Android platform. As described below, I consider the relative changes in the method declarations of the 37 Java APIs copied in Android, as compared to other APIs in the Android core libraries and frameworks. There is much less change, and thus much greater stability, in the 37 Java APIs, compared to other APIs in Android. This suggests that the stability of Android is driven by the 37 Java APIs and their declaring code, and that they are therefore important to Android. This confirms that Google's copying was qualitatively substantial because what Google copied was an important contributor to Android's stability. I conduct alternative stability analyses of the 37 Java APIs to the Android platform, accounting for, and in rebuttal to, arguments made by Dr. Astrachan that 61 classes or the java.lang package are necessary to use the Java programming language. I refer to these analyses as sensitivity analyses. Even when I exclude the 61 classes or the java.lang package, including their declaring code, I find that the remaining portion of the 37 Java APIs exhibited much less change than the other APIs in Android, suggesting that the stability of Android is driven by these 37 Java APIs, and that this code is therefore important to Android. Thus, even under Dr. Astrachan's arguments, Google's copying was qualitatively substantial because what Google copied was important to Android's stability. My full analysis in this regard follows.

a) Stable API packages created a better Android platform

79. One advantage to Google of copying the code and structure, sequence and organization of the 37 Java API packages is that those API packages, in addition to having already been written, would also be expected to be of higher relative quality because they had been available and in use for a long period of time. All else being equal, their relative maturity meant they would be relatively more stable than API packages written from scratch.

b) Past research supports the concept of software platform stability based on API changes

80. Several researchers have analyzed software ecosystems and have assessed the effect of API changes on client applications as a part of understanding stability of a platform. A brief review highlighting some relevant academic literature is provided below.

[34] Raw PageRank scores for each class in Android 5.1.0 can be found in Appendix D.

81. Dig and Johnson (2005) studied the changes in the object-oriented components of APIs by looking at the set of public methods of a class library making up its API[35]. They found that 80% of the code changes that break client-side code are API refactorings, which they define as: "Refactorings are program transformations that change the structure of a program but not its behavior. Refactorings include renaming classes or methods, moving methods or variables between classes, and splitting methods or classes." The paper indicates that "most API changes occur as responsibility is shifted between classes (e.g., methods or fields moved around) and collaboration protocol changes (e.g., renaming or changing method signature)." The paper researched more than 20 types of API changes in five case studies and found that most of the API changes that break the backward compatibility are from types of changes including moved method, moved field, deleted method, changed return type, renamed method, renamed field, renamed class and so on – all related to the method declarations of APIs. Their findings on refactoring also emphasize the importance of the structure of programs in relation to their performance – changes to the structure of the APIs in their study were responsible for 80% of the changes that broke applications.

82. Raemaekers, et al. (2012) note that library (API) stability is an important characteristic for application developers in determining whether to use an existing library or develop the equivalent software themselves, and in choosing which libraries to use when they opt for that solution.[36] Accordingly, they note the need for an objective way to measure stability, and particularly the difficulty in designing stable APIs: "Since requirements keep changing and systems keep evolving over time, designing the correct interface that is stable and backward compatible enough in subsequent releases but also flexible enough to adapt to changing requirements can be challenging." They develop a set of four stability measures and demonstrate their use on a set of Java-based libraries. Smaller values of these metrics indicate greater stability, and are thus preferred.

83. McDonnell et al. (2013) studied API stability and adoption in the Android ecosystem by analyzing empirical API evolution data.[37] They analyzed the behavior of ten active open source applications from a variety of domains. They examined usage of Android APIs by these applications, and particularly focused on issue of timing (lags in adopting changed APIs) and quality (the need for bug fixes in the applications). They report a number of conclusions based on their empirical analysis:

---

[35] Dig, D. and R. Johnson, "The Role of Refactorings in API Evolution," *Proceedings of the 21st IEEE International Conference on Software Maintenance (ICSM '05)*, pp. 389–398, Washington, DC, USA, 2005, IEEE Computer Society.

[36] Raemaekers, S., A. van Deursen, and J. Visser. "Measuring software library stability through historical version analysis," *28th IEEE International Conference on Software Maintenance (ICSM)*, pp.378-387, Sept. 23-28 2012. doi: 10.1109/ICSM.2012.6405296

[37] McDonnell, T. B. Ray and M. Kim. "An Empirical Study of API Stability and Adoption in the Android Ecosystem," *29th IEEE International Conference on Software Maintenance (ICSM)*, pp.70-79, 22-28 Sept. 2013. doi: 10.1109/ICSM.2013.18

- "These results show that though mobile apps are heavily dependent on Android OS and its functionality, developers are hesitant to embrace or fully utilize more recent API features."

- "[T]he pace of client updates is slower for widely used, faster evolving APIs and that developers avoid frequent updates to unstable APIs." "28% of Android references in client code are out-of-date with a median lag time of 16 months."

- "These results show that, in general, the files with more API updates are more prone to bugs. The stronger correlation between API update and bug-fix may explain the slower adoption of new APIs – developers may be skeptic[al] about API adoption as it may introduce bugs."

c) Analysis of API Package stability

84. In order to understand the stability of the copied 37 Java APIs compared to other Android APIs over all version releases in both Android and Java, the publicly available method declarations in the released version of Android and Java SE were analyzed. Following prior research, the focus was on changes in the publicly available method declaration, and the metric used here was the number of method structure changes over the prior API level. Clearly, fewer changes reflect greater stability, so smaller values are better, all else being equal. The analysis was limited to public methods and classes as these are the expressions seen by developers. The data required to perform this analysis was a complete specification of the public packages, classes, and methods contained in both Java and Android's development kits (the Java JDK and Android SDK, respectively). These two sets of specifications are referred to hereinafter as the "Java APIs" and "Android APIs."

85. This metric is essentially the same as the "CEM" metric proposed by Raemaekers, et al. (2012) in their Java research[38].

86. First, each version of the JDK between 1.0.2 and 1.8.0 was manually downloaded from publicly available sources.[39] The JDK contains the source files of the various Java API packages at issue, organized into the appropriate hierarchy.

---

[38] Raemaekers, et al. propose four metrics: "CEM, the amount of change in existing methods", "WRM, the weighted number of removed methods", "RCNO, the ratio of change in new to old methods", and "PNM, the percentage of new methods". WRM was less preferred since the number of removed (deleted) methods is believed to generally be small (e.g. see McDonnell et al. (2013), p. 4). In addition, their proposal to weight the count of removed methods would require data on usage which would not be available on the large scale data set we used. RCNO and PNM require identifying and counting new (added) methods. This has the potential to confound the stability analysis with other factors, like innovation. In addition, any newly added methods in one version then become part of the base comparison case, so any changes to those new methods will be reflected in later comparisons. This is a conservative measurement choice, since not treating additions as changes when they are added means that the resulting measure of change is smaller, tending to show greater, not lesser, stability.

[39] Versions 1.1.3 to 1.8.0 were downloaded from "Oracle JDK Archives," Oracle, http://www.oracle.com/technetwork/java/javase/archive-139210.html. Version 1.0.2 was obtained from "JDK Archives," Taiwan National Central University, http://in.ncu.edu.tw/center5/java/jdk/JDK-1_0_2-win32-x86.exe

87. Next, Oracle's Javadoc tool was used to convert the organization of the source files into an API documentation file in HTML.[40] The HTML documentation file can then be parsed using a custom-written PHP script and manually verified against archived web pages that provide the corresponding information online.[41] Following these verification steps, the Java API documentation for all versions was assembled into a single tabular form to be used in the actual analysis.[42]

88. For the Android APIs a file detailing its complete documentation for each version (or API "Level") is available online on Google's Android Git repositories.[43] The documentation for each API Level between 1 ("Base") and 13 ("Honeycomb_MR2") is published as its own XML document. Documentation for Levels 14 ("Ice Cream Sandwich") to 23 (the most current) are published as plain text files. The XML files were converted into tabular form using the Moor XML to CSV tool.[44] The text files were then parsed using a custom PHP script in a manner similar to those from the JDK.[45] After verifying the results against archived web pages, the results were assembled into their final tabular format.[46]

89. For both the Java APIs and Android APIs, the number of methods changed in each package between two subsequent versions was calculated based on differences between the sets of tabular documentation data generated in the previous step. For example, the number of changes is calculated for the package "java.util" to indicate the number of method changes that exist in this package from Android API level 1 to 2, 2 to 3, and cumulatively up until Levels 22 and 23. The number of method changes is counted based on the list of methods existing in both continuous API levels, which means newly added or deleted methods are not considered as a change in this analysis. For any one particular method existing in both continuous API levels, any change in the method structure[47] is counted as one method change in this analysis.[48] A custom R script was used to calculate the number of method changes for each package between all continuous API levels for both Android APIs and Java SE APIs.[49]

---

[40] *See* Javadoc Tool, Oracle, "http://www.oracle.com/technetwork/java/javase/documentation/javadoc-137458.html"
[41] *See* Appendix E for the full PHP parsing script
[42] *See* Java SE APIs & Documentation, Oracle, http://www.oracle.com/technetwork/java/javase/documentation/api-jsp-136079.html
[43] *See* "Android Framework Classes and Services," GoogleGit, https://android.googlesource.com/platform/frameworks/base
[44] *See* Xml To Csv Conversion Tool, CodePlex," https://xmltocsv.codeplex.com/
[45] *See* Appendix F for the full PHP parsing script
[46] *See* Java SE APIs & Documentation, Oracle, http://www.oracle.com/technetwork/java/javase/documentation/api-jsp-136079.html
[47] Method structure includes method abstract, method synchronization, method exception, method return type, method transient, method volatile, method value, method static, method final, method deprecated, method visibility, and method type. If any of the method structure changes in a method, then the method change is counted as 1.
[48] These include changes in any of the following parameters: method abstract, method synchronization, method exception, method return type, method transient, method volatile, method value, method static, method final, method deprecation or method visibility, and method type.
[49] *See* Appendices G and H, respectively, for the R method-counting scripts used to calculate changes across different versions of the Java SE and Android platforms.

90. The Android SDK is subdivided into three different categories based on placement of APIs in categorical folders—defined by Google—according to function and authorship. These categories are described below.

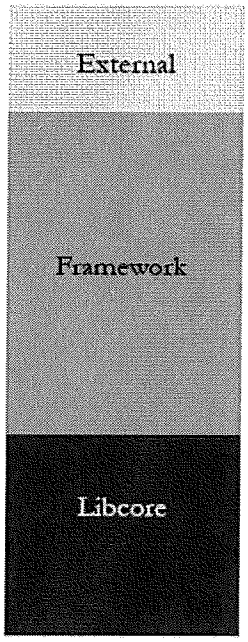
91. Libcore – this category represents the critical core of the Android platform and consists of 64 APIs, including 51 APIs developed by Oracle ("Oracle APIs"), 5 APIs developed by Google ("Google API"), and 8 APIs developed by 3rd parties ("3rd Party APIs"). The 51 Oracle APIs in this category include the 37 infringed APIs.

92. Framework – this category consists of 102 APIs, and consists of 100 Google APIs and 2 Oracle APIs.

93. External – this category consists of 34 APIs, including 2 Oracle APIs and 32 3rd Party APIs.

94. These categories are depicted below in Figure 13.

Figure 13: API Folder Categories of the Android SDK



- **Mostly 3rd Party**
  - 6% Oracle
  - 94% Other
- **Mostly Google**
  - 98% Google
  - 2% Oracle
- **Mostly Java/Oracle**
  - 8% Google
  - 80% Oracle
  - 13% Other

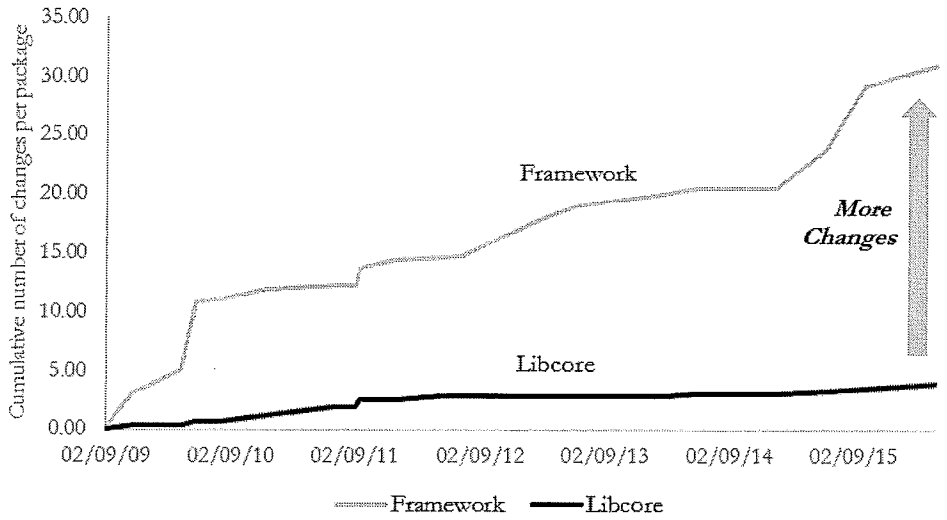
Libcore API package list

android
android.system
dalvik.annotation
dalvik.bytecode
dalvik.system
java.awt.font
java.beans
java.io
java.lang
java.lang.annotation
java.lang.ref
java.lang.reflect
java.math
java.net
java.nio
java.nio.channels
java.nio.channels.spi
java.nio.charset
java.nio.charset.spi
java.security
java.security.acl
java.security.cert
java.security.interfaces
java.security.spec
java.sql
java.text
java.util
java.util.concurrent
java.util.concurrent.atomic
java.util.concurrent.locks
java.util.jar
java.util.logging
java.util.prefs
java.util.regex
java.util.zip
javax.crypto
javax.crypto.interfaces
javax.crypto.spec
javax.net
javax.net.ssl
javax.security.auth
javax.security.auth.callback
javax.security.auth.login
javax.security.auth.x500
javax.security.cert
javax.sql
javax.xml
javax.xml.datatype
javax.xml.namespace
javax.xml.parsers
javax.xml.transform
javax.xml.transform.dom
javax.xml.transform.sax
javax.xml.transform.stream
javax.xml.validation
javax.xml.xpath
org.json
org.w3c.dom
org.w3c.dom.ls
org.xml.sax
org.xml.sax.ext
org.xml.sax.helpers
org.xmlpull.v1
org.xmlpull.v1.sax2

95. The Oracle JDK was subdivided into two different categories based on whether they included the 37 infringed APIs or not.

96. Between two subsequent versions the number of method changes in each package for Android SDK and Oracle JDK, respectively, was calculated, and then an average computed for each category. The cumulative number of changes per package in each category was then graphed to visually track the stability of each version over time.[50] In the Android SDK, the Libcore APIs, which are largely composed of Java APIs, stabilized much earlier than the largely Google Framework APIs, and had far fewer cumulative changes per package across all API levels. The Libcore API can be seen to have stabilized after Android API Level 9, which was released in December, 2010. (See Figure 14, below)

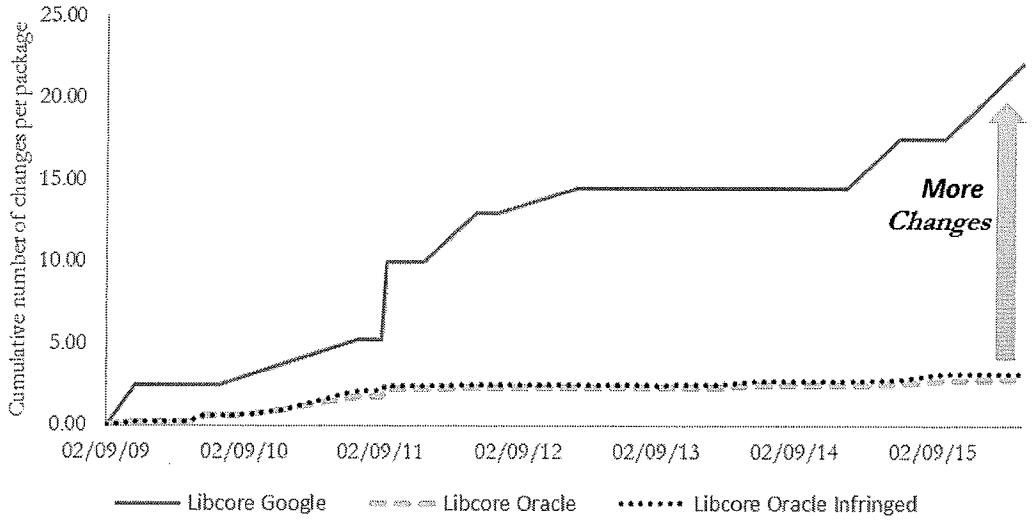
Figure 14: Cumulative Changes per Package (Framework vs Libcore)



97. The more rapid stabilization of the Android Core compared with its Framework can be attributed to the presence of the Oracle APIs. In the Libcore APIs, the Oracle APIs independently achieved stability much earlier as compared to the Google APIs. In fact, the copied 37 Java APIs (which I will refer to here as "the 37 Copied Oracle APIs" or "37 Copied APIs") included in the Oracle APIs represent approximately 73% of the relatively stable Android Core. Moreover, their role in the stabilization of the Android Core is evident when isolating their specific contributions with respect to stability. As illustrated below in Figure 15, cumulative changes for both the 51 Oracle APIs and only the 37 Copied Oracle APIs in the Libcore category tapers off beginning with API level 9, which was released in December, 2010, while Libcore Google APIs have many more cumulative changes per package across all API levels and apparently have yet to stabilize, even in the latest version. This suggests that the stability of the Android core is driven by the 37 Copied Oracle APIs.

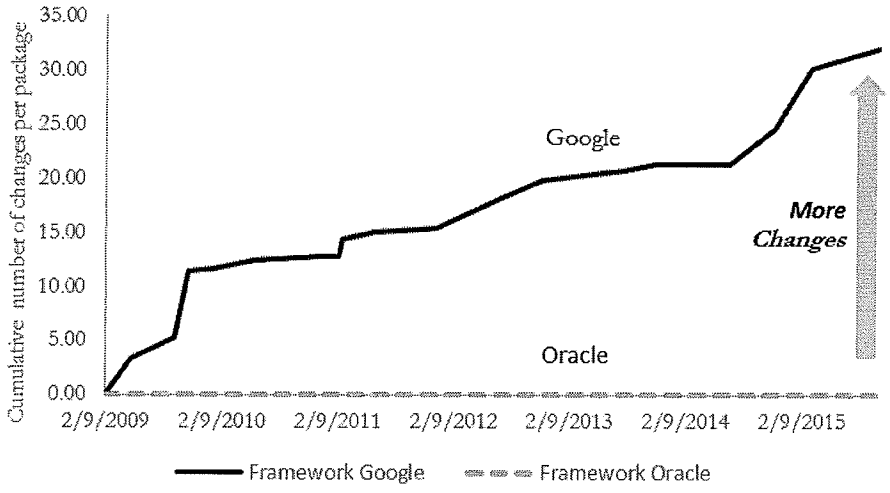
[50] Note that the relatively small number of "External" category APIs are not included in the analysis.

Figure 15: Cumulative Libcore Changes per Package (Google vs. Oracle)



98. The Android Framework, 98% of which is composed of Google APIs, took far longer to stabilize. There are only two Oracle APIs in the Android Framework category, and there is no method change for those two Framework Oracle APIs over all API levels. As a result, essentially 100% of the change volatility in the Android Framework APIs arises from Google APIs. As illustrated in Figure 16, below, the Framework Google APIs have still not demonstrated stabilization, even in the most recent API Level.

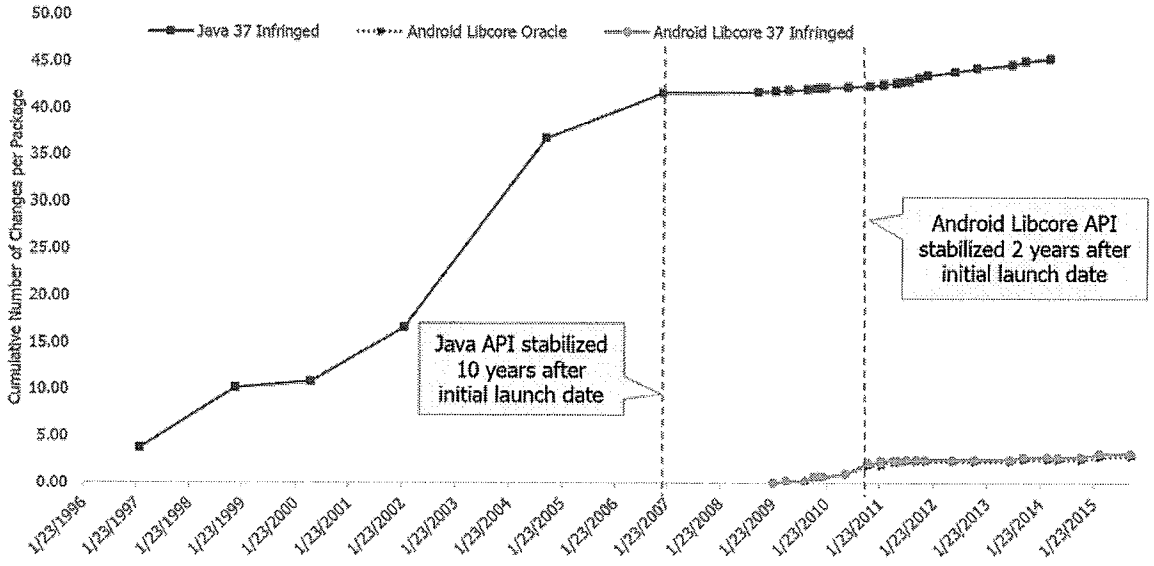
Figure 16: Cumulative Framework Changes per Package (Google v Oracle)



99. The same analysis was conducted for all Java APIs in the JDK. As illustrated below in Figure 17, there were a large number of changes in JDK Java APIs before December, 2006. JDK Java APIs can be seen as

stabilizing around 10.9 years after the first release of the JDK. In contrast, the Android core APIs, which are driven by the 37 copied Java APIs, stabilized around Android API level 11 in February, 2011, only 2.3 years after the first release of Android. Absent the copied Java APIs, it is reasonable to assume that the evolution of the Android core APIs would have exhibited a trajectory more like both the early years of the Java APIs and the early years of the Google-written Framework APIs. It is also reasonable to assume that developers' adoption of Android APIs would have taken a much longer time without the copied Java APIs. This would be consistent with the finding of McDonnell et al. (2013) who concluded that "developers seem hesitant to embrace unstable, fast-evolving APIs quickly".[51]

Figure 17: Copying as an 8-Year Market Head Start[52]



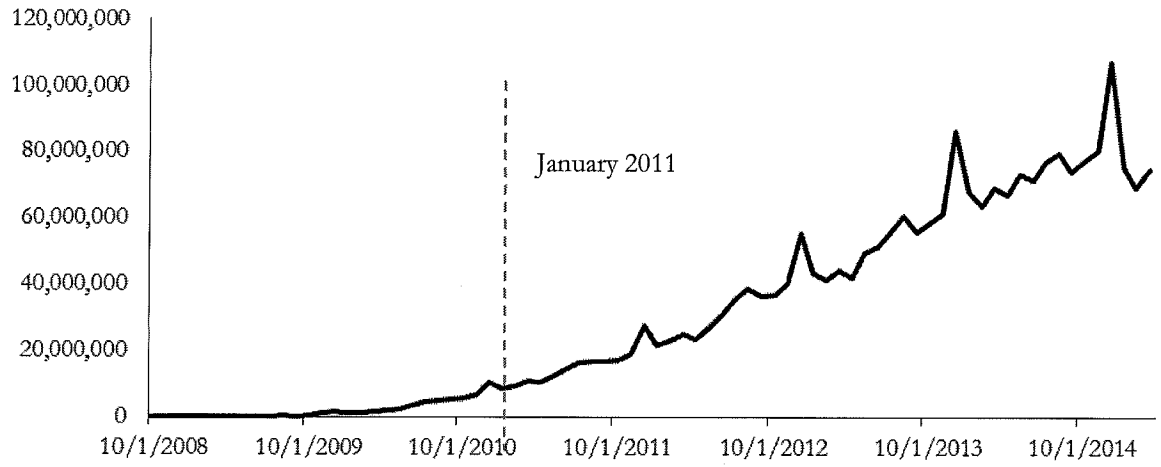
100. The stability of the 37 copied Java API packages contributes to the significant growth of Android. I have observed this growth in three different ways: (1) growth in user activations, (2) growth in the number of active developers and (3) growth of the number of available apps.

101. As I have illustrated below in Figure 18, there is a distinct increase in growth in user activations circa January 2011.

[51] McDonnell, T. et al. (2013)

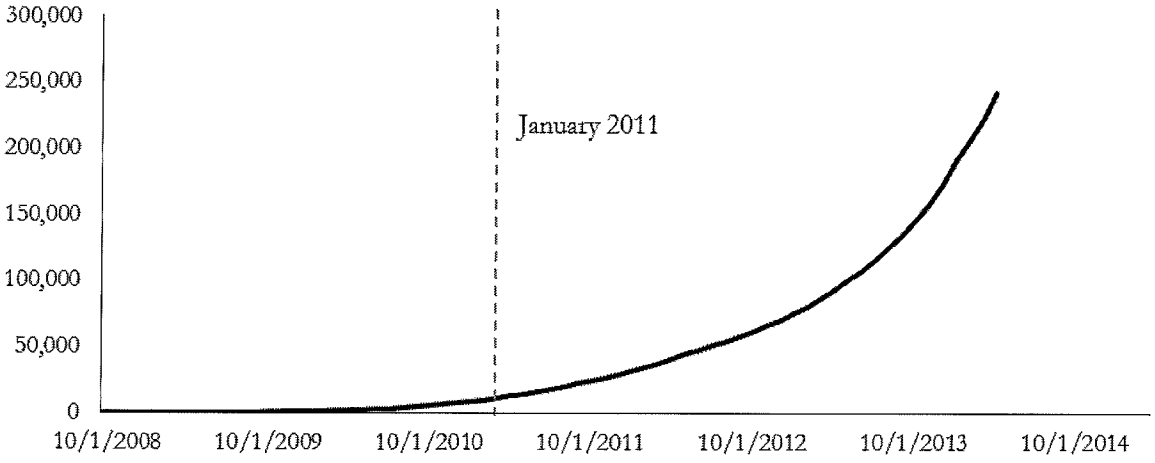
[52] For the curve of the JDK Java APIs, only publicly facing APIs are included since these are the only ones for which the number of changes can be measured. This excludes the following 5 APIs from use in this analysis: javax.crypto, javax.crypto.interfaces, javax.crypto.spec, javax.net, and javax.net.ssl.

Figure 18: Android activations over time[53]



102. As I display in Figure 19, there is distinct growth in the number of active developers circa January 2011.

Figure 19: Android Developer Community growth over time[54]



103. As the two figures above illustrate, Android activations and Android developer community growth, both support the notion that copying of the Java APIs enabled Google to much more quickly release and promote Android than would have otherwise been the case, as well as aiding the acceptance of Android in the developer community. The graphs suggest that, by copying the 37 time-tested Java APIs, Google may have avoided on the order of eight years of what would have otherwise been relative API instability.

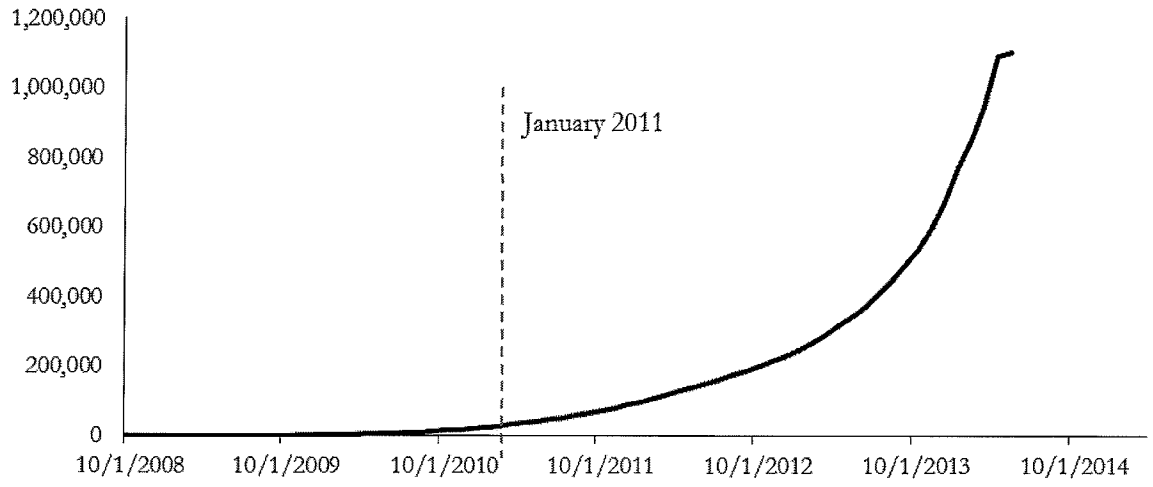
[53] GOOG-00022382

[54] Source: https://github.com/MarcelloLins/GooglePlayAppsCrawler. The date in this analysis is the last release date of the apps in the database

d) Stable API Packages Helped Fuel the Growth of Apps

104. In a similar fashion, the impact of the stability and availability of Java APIs on the success of Android is also supported by the number of apps available for Android. As I display in Figure 20, below, there is distinct growth in the number of apps available on the market circa January 2011.

Figure 20: Android Application availability over time[55]



105. The stability that the 37 Copied Oracle APIs imparted to the Android platform aided the development and adoption of apps on the platform. As discussed further below, adoption of apps on the platform fed the additional growth of the developer community, which ultimately reinforced a positive cycle that benefitted Android and Google. An independent published research paper suggests that the success of applications produced on the Android platform has been linked to the stability and quality of the APIs that are used to create the applications.[56] Low-quality Android apps tend to use Android APIs that have a high change frequency and greater fault proneness. This is because it is difficult for app developers to keep their content working on rapidly changing API versions, which leads to (for example) defects that detract from the end-user experience.

e) Results of the Stability Analysis are Consistent with Established Understandings of How Applications Drive Platforms

106. The results above are consistent with my general understanding of how the availability of applications help to drive platform adoption. I have been studying the adoption of software process innovations and the

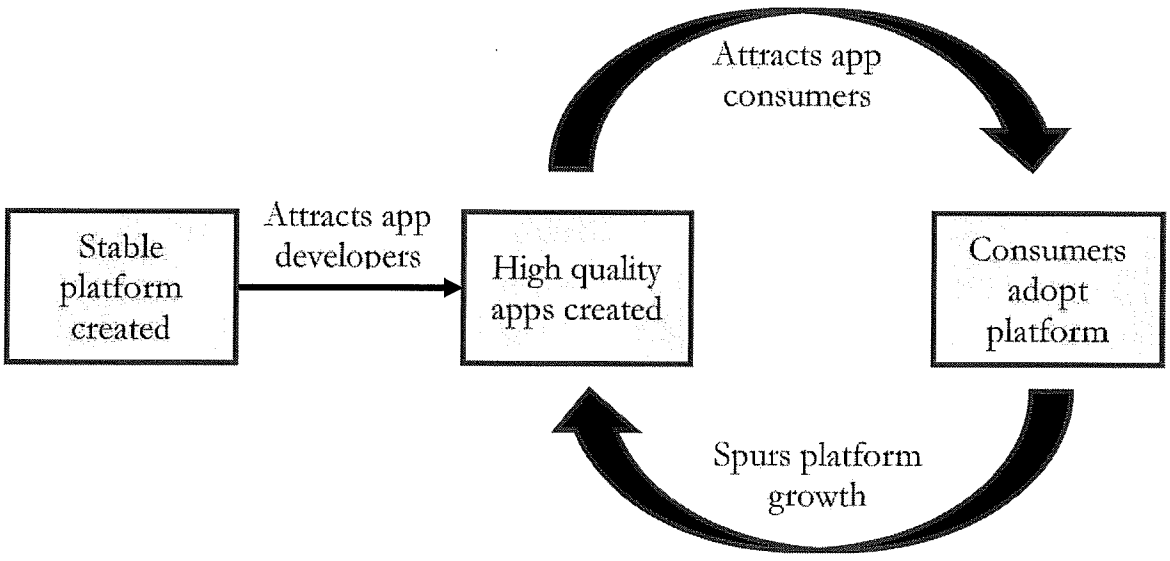
[55] Source: https://github.com/MarcelloLins/GooglePlayAppsCrawler. The date in this analysis is the last release date of the apps in the database

[56] Tian, Y. et al, "What Are the Characteristics of High-Rated Apps? A Case Study on Free Android Applications", *31st IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2015.

measurement of object-oriented software since the early 1990s, resulting in a number of peer-reviewed research publications. This, combined with my research on economic network effects, gives me a clear perspective on the mechanisms by which software innovations become widely adopted among a community of software developers. In general terms this can be captured by the diagram in Figure 21, below. An innovative platform is created and introduced to the market. This platform allows for, and mutually benefits, a set of complementary goods. (Common consumer examples include such items as videogame consoles, which attract videogames as their complementary good.) Once high quality apps are available, this makes consumers more likely to choose a platform that runs these apps, e.g. a videogame console with a popular game (the complementary good) motivates consumers to buy that console. Sales of that console then attract other game developers to write for that console, which then further expands the popularity of the console which commences a positive feedback cycle.

107. The relation of this general phenomenon to the facts in this case is as follows. Here, it is the Java programming platform that initially allows the creation of apps. The Java APIs contributed to the Java platform's widespread adoption by developers. The Android programming platform, based on the Java APIs, was used to create the Android mobile operating system (OS). This OS attracts popular apps (the complementary goods), which, in turn, attract consumers who choose a device which runs that mobile OS so that they can use those apps. The sales of those devices then attract other app developers to also write for that platform, which then further expands the popularity of the platform, which commences a positive feedback cycle, as shown below in Figure 21.

Figure 21: The Positive Feedback Loop of API Stability



108. Google benefited from using the copied code and the structure, sequence and organization of the 37 Java API packages to leverage their popularity and familiarity among developers in order to quickly attract developers to the Android platform when it was first created. In addition to this positive effect, by using the Java APIs Google was able to bring a higher quality Android to market more quickly. There is substantial evidence of this motivation, as is summarized by the Court of Appeals for the Federal Circuit: "… Google wanted to capitalize on the fact that software developers were already trained and experienced in using the Java API packages at issue. The district court agreed, finding that, as to the 37 Java API packages, "Google believed Java application programmers would want to find the same 37 sets of functionalities in the new Android system callable by the same names as used in Java." Copyrightability Decision, 872 F. Supp. 2d at 978. Google's interest was in accelerating its development process by "leverag[ing] Java for its existing base of developers."[57]

109. A senior Android executive testified in deposition that removal of the Java API packages would lead to bugs in applications on the platform: "Q: Do you think that if -- if -- if Google removed the aspects of the Java platform and the Java APIs, in particular, and moved to a new programming environment in which app developers have to adjust to, there would be growing pains?... THE WITNESS: Yes, there would be growing pains. I mean, there's growing pains when we add new APIs and frameworks. … Q Isn't it -- isn't it true that one of the growing pains that Google and -- and Android developers would experience if Google moved away from the Java APIs to a new platform might be bugs in applications? A Yeah, absolutely. We -- we switched our runtime over, and there were growing pains associated with that." (Ghuloum 30(b)(6), Anwar, 150:25-151:17, Dec. 9, 2015)

110. Furthermore, as demonstrated above, the point at which the Android core API begins to fully stabilize, early 2011, coincides approximately with the point at which the number of Android device activations, active developers, and Android apps all begin to accelerate. By August 2010, in a letter to the Google Board of Directors, then CEO Eric Schmidt reflected on early indications of the scale that would later be achieved, noting "For Android, we're now at 200,000 device activations per day, which is a 60% month-over-month growth rate. You start calculating what that will be in a year … and it looks to me as though Android is well past escape velocity at every level." (Google 26-00025769 at 770).

111. Long-term trends in the mobile industry at large clearly demonstrate that an active marketplace of high-quality apps is an important driving factor of that platform's success. Nokia and BlackBerry mobile phones previously held strong positions in the mobile space, with roughly 67% market share in 2009 – around the same time when the app community for the iOS and Android platforms began to grow much faster than those for Nokia and BlackBerry.[58]

---

[57] United States Court of Appeals for the Federal Circuit, Oracle America Inc. v. Google Inc. 2013-1021, -1022 p. 51.
[58] Localytics (June 18, 2009), https://www.localytics.com/smartphone-os-wars-develop-for-which-platforms-part-I.

112. Since then, Nokia and BlackBerry's market shares have dropped to less than 3% and their presence in the mobile phone space has been all but eliminated. One of the contributing factors to this decline was the relative dearth of BlackBerry apps measured against their competitors. Windows Phone devices' correspondingly weak market position and poor app availability also serve to corroborate this notion. These industry examples stress the important role of the app developer community in bolstering the health of a mobile platform.

113. An Android executive has recently testified that users are attracted to Android because of the number of available applications:

> "Q Isn't it true that one -- one reason that users are attracted to a device platform is the number of applications available on the platform?... THE WITNESS: I believe that is a factor, yes. ... Q And so the facility and speed with which a platform like Android can attract application developers, in turn, feeds the number of users?... THE WITNESS: I -- I think it can. I think, though, there are other things that -- well, you're asking me specifically about whether application availability and the speed of application availability on the platform attracts users? ... Q Correct. A Yeah, I believe it does have an influence on -- on this. Q And so if suddenly there were -- were less applications available for a platform, that would have some impact on the appeal of the platform, the device platform to users?... THE WITNESS: Yeah. If suddenly, you know, the applications went away or a significant number of applications went away, especially certain, you know, top 1,000 applications went away, that would be problematic. I think that's unlikely to happen, but, yeah, that would be a problem." (Ghuloum 30(b)(6), Anwar, 149:15-150:23, Dec. 9, 2015)

f) API Stability Builds Application Performance, the Developer Ecosystem and End User Engagement

114. Oracle provided the Java APIs, which permit app developers to invoke prewritten code. Once a Java API has been created, any developer with access to it can make full use of the functionality by invoking associated prewritten code from the APIs.

115. All else being equal, APIs that have been in existence longer are more likely to have been debugged, enhanced, and generally improved until they become relatively mature. As their performance becomes validated and trusted through developer use, the functionality they provide becomes more stable and increasingly useful in producing applications.

116. Newly released APIs have existed for a shorter period, which means they are still subject to regular changes and updates as defects are fixed. New APIs are thus seen as relatively unstable, since the functionality they provide is more likely to be error-prone; it is more difficult to make software function reliably with an API that is constantly changing.

117. Furthermore, since stable APIs produce high-quality apps that attract more users, the growing user base in turn attracts more developers to the platform as well. This catalyzes a positive feedback loop that drastically increases the growth rate of both the user and developer communities.

g) Sensitivity Analysis of the Stability results under Google's arguments

118. As discussed earlier in this report, Dr. Astrachan asserts that 61 particular classes or the java.lang package are necessary to use the Java programming language.[59] As discussed in Dr. Schmidt's Rebuttal Report at ¶¶ 244, 245 and 264, Dr. Astrachan's assertions in this regard are flawed. However, in the interest of completeness and in response to Dr. Astrachan's arguments, I have conducted the stability analysis discussed above in a manner that excludes the 61 classes and the java.lang package, respectively, in order to understand the importance of the classes of the 37 Java APIs outside of the material that Dr. Astrachan asserts is necessary to the language. I refer to these analyses as sensitivity analyses, because they determine how sensitive the stability analysis results are to exclusion of the 61 classes or the java.lang package. The results of sensitivity analysis excluding the 61 classes is set forth first below, followed by sensitivity analysis excluding the java.lang package. As demonstrated below, even when the 61 classes or java.lang package are excluded, this does not materially change the results set forth in the stability analysis above, which therefore reinforces the following conclusions:

- The stability of the Android platform is driven by the stability of the 37 copied Java API packages and the declaring codes and SSO within;
- Copying the 37 infringed Java API packages provided Google with an 8-year head start in developing the Android platform;
- The stability of the 37 copied Java API packages contributes to the significant growth of Android.

119. I expect that the declaring code and structure, sequence and organization of the 37 Java API packages copied by Google, would be of higher relative quality because they had been available, and in use, for a longer period of time. All else being equal their comparative maturity meant they would be relatively more stable than API packages written from scratch.

120. The foregoing analysis was carried out by removing the 61 Java classes or the java.lang package, pursuant to Dr. Astrachan's arguments, in both Android SDK and Java JDK. In the Android SDK, the Libcore APIs, which are largely composed of Java APIs, stabilized much earlier than the largely Google Framework APIs, and had far fewer cumulative changes per package across all API levels, even accounting for each of Dr. Astrachan's arguments.

[59] Astrachan Report, ¶¶ 163-164

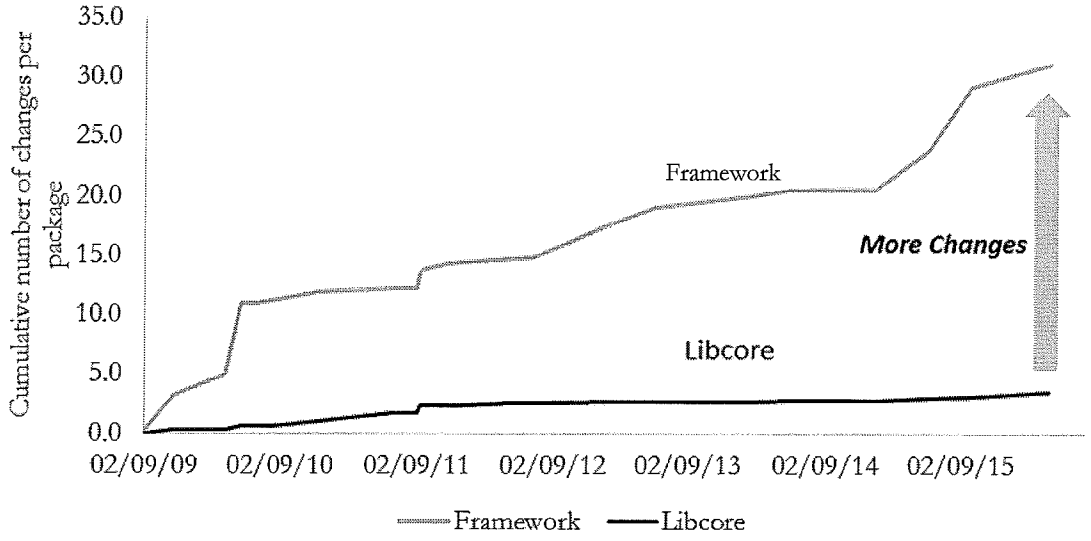
121. Even when we exclude the 61 classes or the java.lang package we find that the remaining portion of the 37 Java APIs exhibited much less change than the other APIs in Android, suggesting that the stability of Android is driven by these portions of the 37 Java APIs, and that this material is important to Android. Thus, even under Dr. Astrachan's argument, my analysis confirms that Google's copying was qualitatively substantial because what Google copied was so important to Android's stability. The results of my sensitivity analysis are set forth below.

(i) 61 Class Sensitivity Analysis

122. The first sensitivity test involved removing the 61 Java classes listed in Exhibit 1062 from both the Android SDK and the Java JDK in the original datasets for the stability analyses. The same analyses were then rerun for both Android SDK and Java JDK.

123. Figure 22, below, shows the cumulative number of changes per package in Android SDK over time for the Android Libcore and Framework APIs in this scenario. In the Android SDK, the Libcore APIs, which are largely composed of Java APIs, stabilized much earlier than the largely Google Framework APIs, and had far fewer cumulative changes per package across all API levels. The Libcore API can be seen to have stabilized after Android API Level 9, which was released in December, 2010.

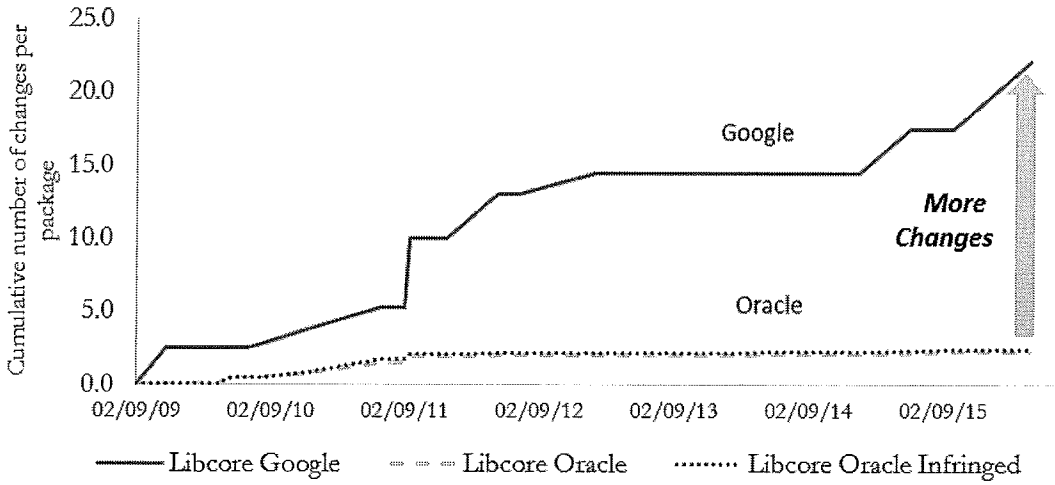
Figure 22: Cumulative Changes per Package (Framework vs Libcore) for the 61 Class Scenario



124. The more rapid stabilization of the Android Core compared with its Framework can be attributed to the presence of the Oracle APIs. In the Libcore APIs, the Oracle APIs independently achieved stability much earlier as compared to the Google APIs. As illustrated in Figure 23, below, cumulative changes for both the

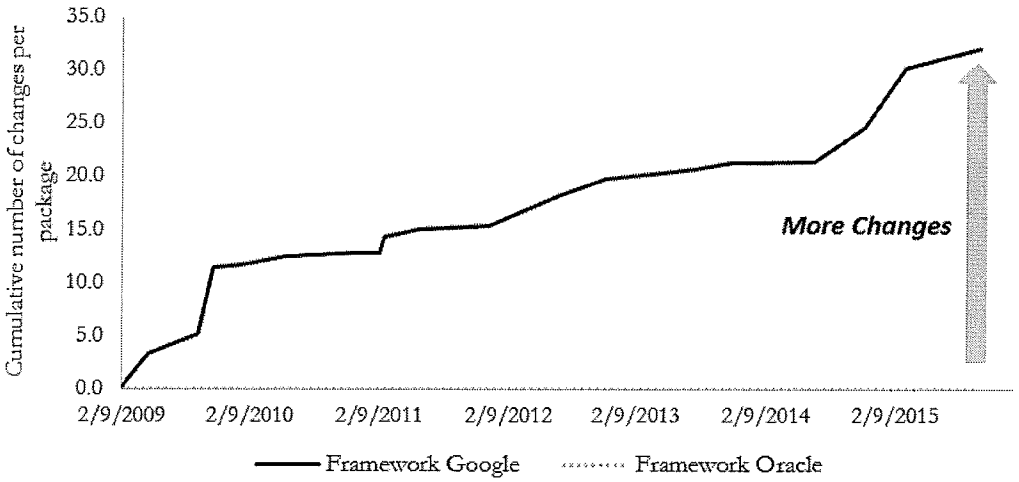
51 Oracle APIs and only the 37 copied Oracle APIs in the Libcore category taper off beginning with API level 9, which was released in December, 2010, while Libcore Google APIs have many more cumulative changes per package across all API levels, and have yet to stabilize, even in the latest version. This suggests that the stability of the Android core is driven by the declaring code and SSO of the 37 copied Oracle APIs.

Figure 23: Cumulative Libcore Package Changes per Package (Google vs. Oracle)–61 Class Scenario



125. The Android Framework, 98% of which is composed of Google APIs, took far longer to stabilize. There are only two Oracle APIs in the Android Framework category, and there are no method changes for those two Oracle APIs across all API levels. As a result, essentially 100% of the change volatility in the Android Framework APIs arises from Google APIs. As illustrated below in Figure 24, Framework Google APIs have apparently still not achieved stabilization, even in the most recent API Level.

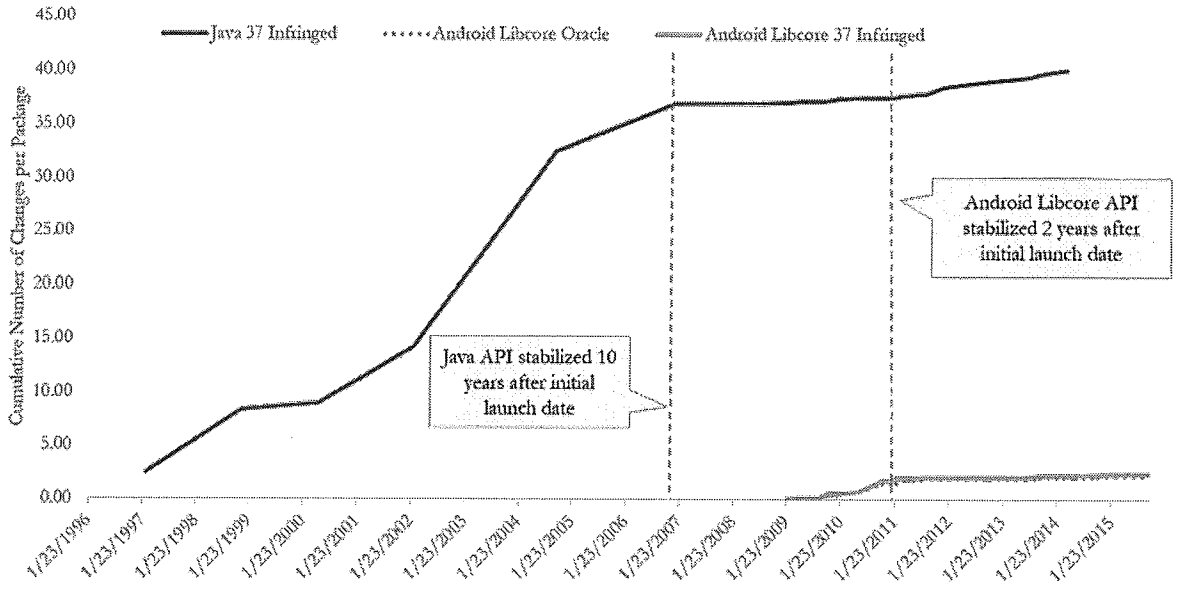
Figure 24: Cumulative Framework Changes per Package (Google v Oracle) for 61 Class Scenario



126. The same analysis with the removal of the 61 classes was conducted for all Java APIs in the JDK. As illustrated below in Figure 25, there were a large number of changes in the JDK Java APIs before December, 2006. JDK Java APIs can be seen stabilizing around 10.9 years after the first release of the JDK. In contrast, the Android Core APIs, which are driven by the 37 copied Java APIs, stabilized around Android API level 11 in February, 2011, only 2.3 years after the first release of Android. It is also reasonable to assume that developers' adoption of Android APIs would have taken a much longer time without the copied Java APIs. This would be consistent with the finding of McDonnell et al. (2013) who concluded that "developers seem hesitant to embrace unstable, fast-evolving APIs quickly".[60]

[60] McDonnell, T. et al. (2013)

Figure 25: Copying as an 8-Year Market Head Start - 61 Class Scenario



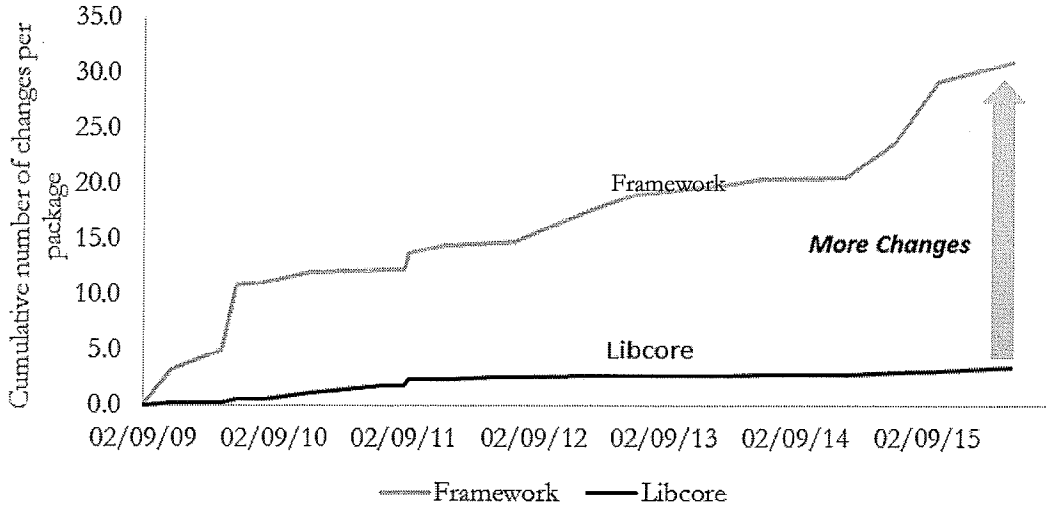
127. In summary, this sensitivity analysis demonstrates that the removal of the 61 classes in Exhibit 1062 had no substantive effect on the stability analysis results. This is because the removal of these classes would only reduce the number of method changes across subsequent versions of JDK and Android Libcore APIs.

(ii) Java.lang Package Sensitivity Analysis

128. The second sensitivity analysis involved removing the java.lang package from both Android SDK and Java JDK in the original datasets for the stability analyses. The analyses were then rerun for both Android SDK and Java JDK.

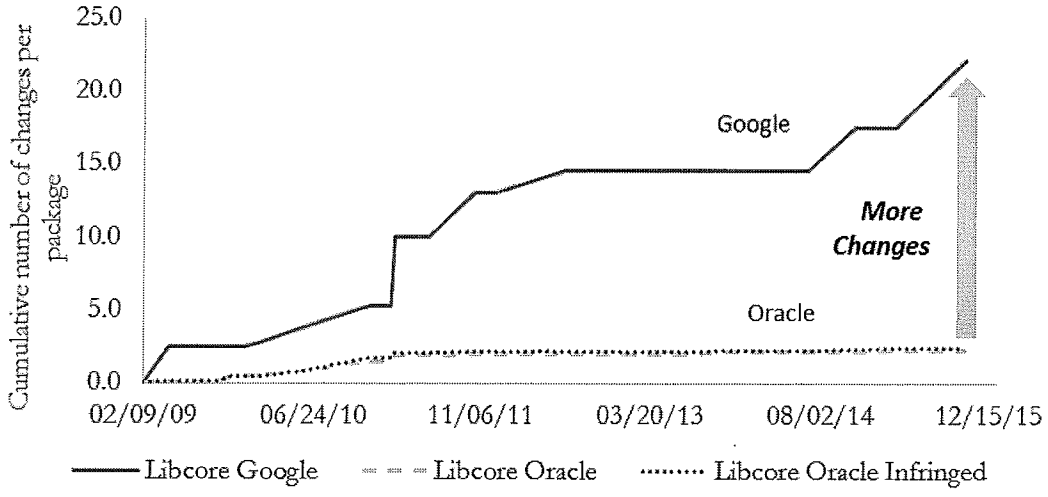
129. Figure 26, below, shows the cumulative number of changes per package in Android SDK over time for Android Libcore and Framework APIs in this scenario. In the Android SDK, the Libcore APIs, which are largely composed of Java APIs, stabilized much earlier than the largely Google Framework APIs, and had far fewer cumulative changes per package across all API levels. The Libcore API can be seen to stabilize after Android API Level 9, which was released in December, 2010.

Figure 26: Cumulative Changes per Package (Framework vs Libcore) for Java.lang Scenario



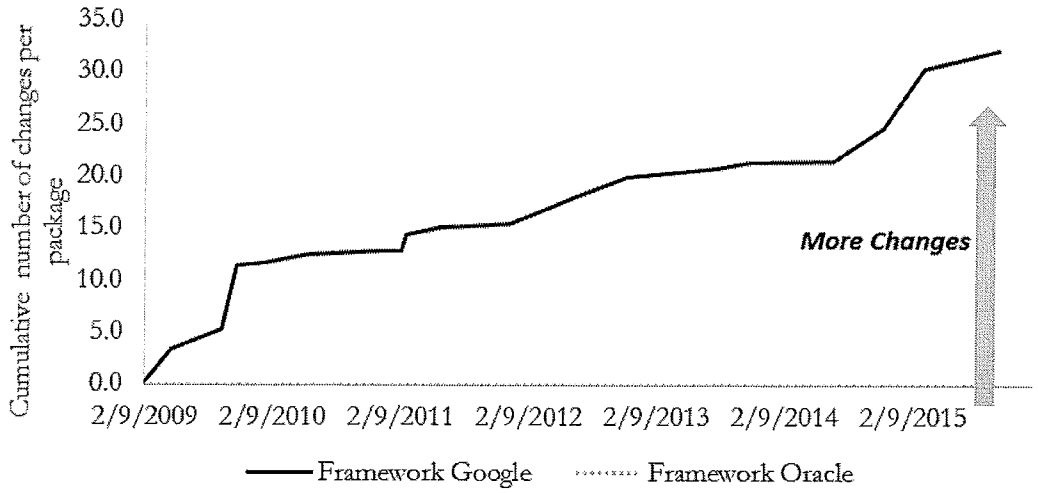
130. The more rapid stabilization of the Android Core compared with its Framework can be attributed to the presence of the Oracle APIs. In the Libcore APIs, the Oracle APIs independently achieved stability much earlier as compared to the Google APIs. As illustrated in Figure 27, below, cumulative changes for both the 51 Oracle APIs and only the 37 copied Oracle APIs in the Libcore category taper off beginning with API level 9, which was released in December, 2010, while the Libcore Google APIs have many more cumulative changes per package across all API levels, and have apparently yet to stabilize, even in the latest version. This suggests that the stability of the Android core is driven by the declaring code and SSO of the 37 copied Oracle APIs.

Figure 27: Cumulative Libcore Package Changes per Package (Google vs. Oracle)–Java.lang Scenario



131. The Android Framework, 98% of which is composed of Google APIs, took far longer to stabilize. There are only two Oracle APIs in the Android Framework category, and there were no method changes for those two Oracle APIs across all API levels. As a result, essentially 100% of the change volatility in the Android Framework APIs arises from the Google APIs. As illustrated below, in Figure 28, Framework Google APIs have apparently still not achieved stabilization, even in the most recent API Level.

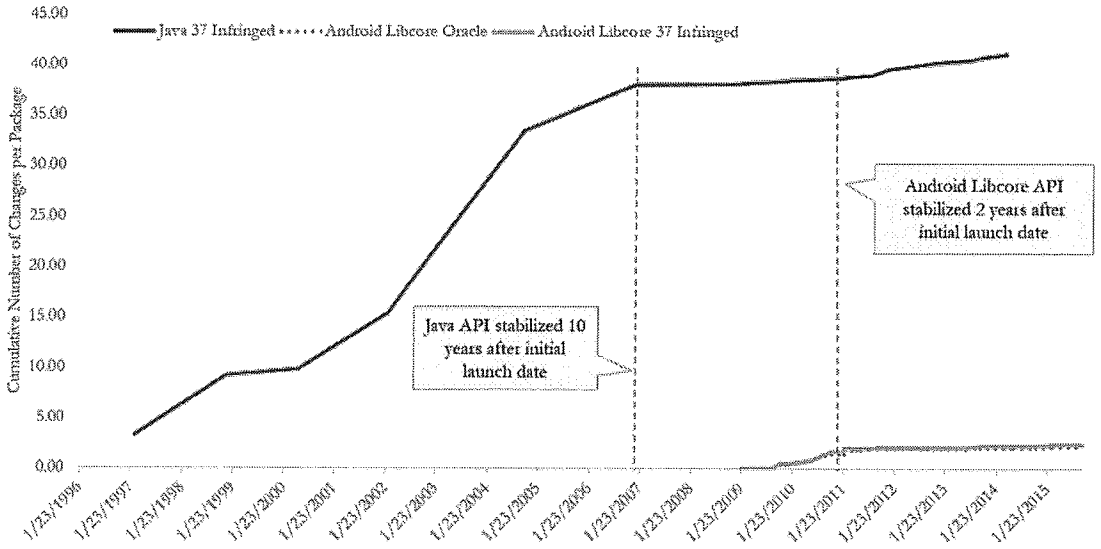
Figure 28: Cumulative Framework Changes per Package (Google v Oracle) for Java.lang Scenario



132. The same analysis without the entire java.lang package was conducted for all Java APIs in the JDK. As shown below in Figure 29, there were a large number of changes in JDK Java APIs before December, 2006. JDK Java APIs can be seen stabilizing around 10.9 years after the first release of the JDK. In contrast, the Android core APIs, which are driven by the 37 copied Java APIs, stabilized around Android API level 11 in February, 2011, only 2.3 years after the first release of Android. Absent the copied declaring code and SSO Java APIs, it is reasonable to assume that the evolution of the Android core APIs would have exhibited a trajectory more like both the early years of the Java APIs and the early years of the Google-written Framework APIs. This would be consistent with the finding of McDonnell et al. (2013) who concluded that "developers seem hesitant to embrace unstable, fast-evolving APIs quickly".[61]

[61] McDonnell, T. et al. (2013)

Figure 29: Copying as an 8-Year Market Head Start for Java.lang Scenario



C. Copying 11,000 Or 7,000 Lines of Code and the Structure, Sequence and Organization of the 37 API Packages is a Substantial Taking

1) 11,000 Or 7,000 Lines Reflecting The SSO Of The 37 API Packages Is A Large Amount And Substantial Body Of Code

133. Google copied 11,000 lines of code, as described in my initial report, relying on the analysis of Mr. Zeidman. Dr. Astrachan asserts that Google copied 7,000 lines of code. It is my opinion that 11,000 lines of code, or even 7,000 lines of code, is a large and substantial amount of material to copy from one body of programs (the 37 API packages in Java) into another body of programs (the 37 API packages in Android). Objectively, the volume of code copied is large and copying was clearly systematic, and not merely incidental. The fact that this code reflects the structure, sequence and organization of the API packages also demonstrates that the quantity taken was substantial.

134. In the software industry generally, and in my research and professional experiences measuring the value and significance of software, I have observed that software that contains amounts of code on this order of magnitude can contribute the bulk of value to an overall program or software system. For example, the Apollo mission Lunar Module system is reported to have contained only about 10,000 lines of code in a context where lives depended on the reliability and effectiveness of that code.[62] The first releases of the entire UNIX operating system also were reported to have been comprised of on the order of 10,000 lines of code, and there is wide

[62] Capture of Apollo Lunar Module Reliability Lessons Learned: Reliability Engineering, NASA, http://llis.nasa.gov/lesson/1835 (accessed Feb. 8, 2016)

agreement that UNIX was a significant advance in computing history.[63] A completely computerized Canadian nuclear power plant shutdown system is reported to contain about 6,000 lines of code, demonstrating that even code of the approximate volume cited by Dr. Astrachan in this case can have an extraordinary impact.[64] Thus, quantitatively, an amount of code similar to that at issue in this matter can be extremely valuable in terms of how central, impactful and effective the lines of code are to the system of which they are a part.

135. Google itself recognized that the material that it copied was important to Android, and therefore was a taking of a substantial portion of Java SE. In June 2006, after negotiations with Sun had failed, Google recognized that Android's Java class libraries were "half-ass at best" and that it "need[ed] another half of an ass."[65] But instead of creating or licensing, Google copied Sun's 37 Java APIs. Google took these APIs because they were "the good stuff."[66] There was a large community of developers who use the Java APIs to develop applications.[67] And Google knew that copying the Java APIs would entice these developers to join Android.[68] Android's lead virtual machine engineer, Daniel Bornstein, indicated that Google chose these APIs because developers know them well.[69] Google engineers also especially appreciated the safety features these Java APIs provided.[70] Finally, the carriers and the OEMs trusted and supported the Java APIs.[71]

2) The Astrachan Report Undercounts and Undervalues the Declaring Code and the Structure, Sequence and Organization of the 37 API Packages in Both Java and Android

136. Dr. Astrachan's report repeatedly attempts to break down the declaring code and the structure, sequence and organization of the copyright protected material at issue, in order to artificially focus only on individual components of the code. He ignores the creative combinations of code elements that result in highly expressive lines of code, and further ignores the creative choices available in which to intricately organize the lines of code, methods, classes and packages at issue. Through this artificial mode of analysis Dr. Astrachan improperly

---

[63] History of UNIX, R. Hauben, http://minnie.tuhs.org/TUHS/Mirror/Hauben/unix.html, http://minnie.tuhs.org/TUHS/Mirror/Hauben/unix-Part_II.html ("The Edition 6 UNIX code contained less than 10,000 lines, which positioned it nicely to become the first really accessible operating system.")
[64] Leveson, N.G., "High-pressure steam engines and computer software," in *Computer*, vol.27, no.10, pp.65-73, Oct. 1994, *available at* http://sunnyday.mit.edu/papers/steam.pdf, at p. 4
[65] TX 215
[66] GOOGLE-40-00003728; GOOGLE-02-00298870 (discussion between two Android engineers: "The *only* sense of value I have in this project is in Java stuff we're giving away, not in the product we'll end up shipping.")
[67] TX 158 ("Fact…6M Java developers worldwide. Tools and documentation exist to support app development without the need to create a large developer services organization."); TX 1 ("[e]xisting pool of developers and applications")
[68] TX 158 ("Supporting Java is the best way to harness developers"; "Strategy: Leverage Java for its existing base of developers.")
[69] GOOGLE-02-00384174 ("it behooves us to expose a non-jolting familiar-looking Java API for this functionality, which would mean using standard Java classes and interfaces throughout")
[70] TX 23 ("java is safer"); TX 7 ("Java has a suitable security framework"); TX 1 ("[s]afe sandbox for 3rd party developers")
[71] "The wireless industry ha[d] adopted Java, and the carriers require[d] its support." TX 158; TX 7 ("carriers require managed code"); TX 15 ("Java dominates wireless industry"; "Carriers require Java"). Google knew this and promised its partners that Android will contain "Standard Java class libraries," GOOGLE-24-00010460 (April 2007 Sprint pitch); GOOGLE-24-00019558 (April 2007 Vodafone pitch); GOOGLE-24-00015413 (May 2007 Orange pitch)

undervalues the creativity of the material at issue and creates the artificial perception that only individual code components are at issue. This approach is flawed and inaccurate.

137. The Court of Appeals has already warned that the mode of analysis of breaking down the Java code at issue into individual, constituent elements (yet ignoring their arrangement and organization into a broader creative software work), is a flawed approach because it artificially conceals the true creative value, substantiality and significance of the copyrighted works. The Court of Appeals warned against "dissecting the individual lines of declaring code at issue into short phrases" as that fails "to recognize that an original combination of elements can be copyrightable."[72] The Court of Appeals continued:

> "By analogy, the opening of Charles Dickens' A Tale of Two Cities is nothing but a string of short phrases. Yet no one could contend that this portion of Dickens' work is unworthy of copyright protection because it can be broken into those shorter constituent components. The question is not whether a short phrase or series of short phrases can be extracted from the work, but whether the manner in which they are used or strung together exhibits creativity. Although the district court apparently focused on individual lines of code, Oracle is not seeking copyright protection for a specific short phrase or word. Instead, the portion of declaring code at issue is 7,000 lines, and Google's own "Java guru" conceded that there can be "creativity and artistry even in a single method declaration." Joint Appendix ("J.A.") 20,970. Because Oracle "exercised creativity in the selection and arrangement" of the method declarations when it created the API packages and wrote the relevant declaring code, they contain protectable expression that is entitled to copyright protection."[73]

Yet, despite this admonition, this is exactly the type of argument that Dr. Astrachan attempts to make.

138. Dr. Astrachan compares the lines of copied declaring code to "the entire Android source code." This mode of analysis also ignores that the most important part of the Java packages taken in Android are the declaring code and their organization, which contribute to the stability of Android in a disproportionately significant way, as discussed above. In fact, as discussed above and in the Opening Report of Dr. Schmidt, if the declaring code is removed from the entire Android source code, Android will not even compile on mobile phones. This demonstrates the substantiality of the copied code in the context of the overall Android code. Dr. Astrachan conducts no such substantiality analysis

139. Dr. Astrachan compares the lines of copied declaring code in Android to the overall code of the 37 API packages in Java, but, in doing so, ignores that the copied lines of code are the most central and therefore important part of the Java packages taken, as can be seen by the PageRank analysis above. Dr. Astrachan conducts no such centrality analysis.

[72] United States Court of Appeals for the Federal Circuit, *Oracle America Inc. v. Google Inc.* 2013-1021, 1022, p. 34-35
[73] United States Court of Appeals for the Federal Circuit, *Oracle America Inc. v. Google Inc.* 2013-1021, 1022, p. 34-35

140. Dr. Astrachan compares the number of lines of copied declaring code to the overall code in the Android (Gingerbread) Runtime Core Libraries. This also ignores that the most important part of the Java packages taken by Google are the declaring code and organization, which contributed to the overall stability of Android, as established by the stability analysis set forth above. In particular, the 37 Java APIs contribute to the stability of Android and the overall set of packages in the Android core libraries. The new API packages that Google added to the 37 Java APIs do not contribute the same level of stability to Android. Dr. Astrachan conducts no such stability analysis.

141. Again, the Court of Appeals has criticized this "percentage" approach. The Court noted that it is inappropriate to compare the amount of copied material to the total amount of the infringing work (here Android)[74] since this percentage is under the control of the alleged plagiarist. Further, in considering Google's argument with reference to nine (9) lines of copied code being an "infinitesimal" percentage of the whole, the Court opined that this argument is "without merit" as even a very small percentage of lines can be qualitatively significant[75].

142. Ignoring the substantiality of the 37 Java APIs to the Java and Android APIs, and to the Java and Android platforms as a whole, Dr. Astrachan engages in a misleading "percentage" exercise, comparing the number of lines of copied code to the number of lines of code in the entirety of the APIs in the Java platform, the entirety of the APIs in Android, and the overall Android code base.

143. Furthermore, even if the percentage of the number of lines of code copied were a meaningful measure in this context, Dr. Astrachan undercounts the lines of copied code. Dr. Astrachan asserts that "7,000" lines of code were copied.[76] In fact, as documented in the report of Mr. Zeidman, there are over 11,000 lines of code copied.[77]

144. Also, as noted by Dr. Schmidt, Dr. Astrachan's comparison of the lines of declaring code to implementing code (in which he includes blank lines of code and comments) is not particularly pertinent to understanding the substantiality of the declaring code itself.[78] This is particularly true given that the implementing source code is subordinate to the declaring code, as discussed in the report of Dr. Schmidt, at ¶

---

[74] United States Court of Appeals for the Federal Circuit, *Oracle America Inc. v. Google Inc.* 2013-1021, 1022, p. 58.
[75] United States Court of Appeals for the Federal Circuit, *Oracle America Inc. v. Google Inc.* 2013-1021, 1022, p. 66
[76] Astrachan Opening Report, ¶¶ 140, 224
[77] Astrachan Opening Report, ¶¶ 140, 224. Beyond that, Dr. Astrachan compares the lines of copied Java code to the lines of code in Android, which, as discussed, is generally an inappropriate approach for considering factor 3 of the fair use analysis. It is particularly inappropriate to compare the copied lines of code to the "entire Android source code" because that would include millions of lines of source code of in the Linux kernel. It is inappropriate to consider the Linux kernel to be part of the entire Android source code because was derived from Linux (not independently created) and it could not even be combined with the rest of Android under any circumstances, because Linux is subject to the GPL license which is incompatible with the Apache license that Google appears to be claiming applies to the rest of Android.
[78] Schmidt Fair Use and Rebuttal Report, §VIII-A. I have reviewed the report of Dr. Schmidt, and incorporate and rely upon Dr. Schmidt's counting analysis of the code at issue.

256. Additionally, Dr. Astrachan's focus on comparing the quantity of copied code to all of the underlying Android source code is misplaced because that is not what developers look at. Rather, developers only perceive the expressive declaring code when they are developing apps. Hence, what Google copied is disproportionately more important than the underlying Android source code that they do not perceive. If a percentage approach was deemed to be a useful approach, then the appropriate calculation would to compute the ratio of declaring code copied in Android to the total lines of declaring code in the 37 API packages in Java SE 5. This calculation reveals that declaring code for 624 out of 1057 classes were copied, which represents 59% of the relevant class declarations in Java SE 5.[79]

VII. Other Considerations in Rebuttal: Commercial Expectations Regarding APIs

A. Control over the Java APIs and other Software Development Tools is Known and Expected in the Software Industry and Creates Incentives for Investment In and Creation of New Software

145. Throughout their expert reports, Dr. Astrachan and Dr. Cattell repeatedly argue, in a general way and seemingly independent of any particular fair use factor, that there was a broad expectation across the industry and among developers that all APIs, including the 37 Java APIs at issue, could be freely copied and used, without limitation, restriction or control.[80] Each argues that this assertion is purportedly material to fair use. I do not agree with this factual analysis or the conclusion. As developed further below, I establish that control over, and restrictions on, the copying and use of software generally, and APIs and similar programs in particular, have long been an expected and understood reality in the software industry. I discuss that such control and restrictions on copying and use of software generally and APIs in particular, creates incentives to invest in the creation of such software, and therefore furthers the economic policies that motivate copyright law. I provide historical examples of control over APIs, evidence that the software industry has long understood that APIs, and the Java APIs in particular, are subject to restrictions on copying and use, as well as evidence that control over copying and use of APIs is expected in the burgeoning "API economy" and incentivizes creation in this economy. I also provide evidence that Google itself places numerous limitations on copying and use of its own APIs. These facts weigh against a finding of fair use and rebut Dr. Cattell's and Dr. Astrachan's assertions regarding industry or developer expectations.

B. History and Context Regarding Legal and Practical Control of Software

146. The principle of copyright has existed for centuries and has served society well by protecting authors and thereby encouraging both the production and dissemination of creative works[81]. Information goods, including

[79] Schmidt Fair Use and Rebuttal Report, ¶252
[80] Cattell Opening Report, ¶¶ 35-53; Astrachan Opening Report, ¶¶ 42-95, 171-173, 271-280
[81] Landes, W. and R. Posner, "An Economic Analysis of Copyright Law", 18 *J. Leg. Stud.* 325, 325-33, 344-53 (1989) http://cyber.law.harvard.edu/IPCoop/89land1.html (accessed Feb. 8, 2016)

computer software, have special economic characteristics that differentiate them from many other kinds of commercial products. They generally exhibit high fixed costs to develop the first copy, but generally very low costs of reproduction. This makes them tempting targets for illegal copiers, since such copiers can avoid nearly all the costs of production. In addition, the initial creator also bears the risks of market acceptance - the uncertainty of demand for a creative product. The illegal copier can wait until a creative work has demonstrated that it has an audience, and then choose to only copy the popular works, thus avoiding any market risk.

147. Copyright laws solve this problem by rewarding authorship with a limited time monopoly to produce copies of the copyrighted work. This allows the creator to generate enough total revenue to compensate them for the significant fixed costs and the market demand risk. Society benefits both from the increased production of creative works, and from their increased dissemination, since information goods are, by their nature, inherently public goods which must be disseminated to reach their audience. Without protection, the incentives to both create and disseminate would be reduced. In addition to the primary loss to society of these goods, it is easy to forecast the later loss of downstream creative works that are inspired by earlier output.

148. The digital age has exacerbated the problem of illegal copying, since digital copies, a pattern of ones and zeros, are generally indistinguishable from their originals, whereas in the analog era some protection was afforded to originals when copies could be identified as inferior goods. Digitization has also greatly reduced the cost of copying and the speed of both copying and disseminating the copied work. The growth of the Information Age in general has increased the market size for information goods, which increases both the risks and rewards of content creation. A whole host of business models have evolved to generate revenue from digital information products.

149. Software is clearly an important digital information good in today's commercial markets, and exhibits all of the classic behaviors described above. Production of the first working copy of a piece of software is expensive, as talented software developers are scarce, and thus well-compensated, and software development is difficult to do well, and therefore risky. Its inherently digital nature means that it is subject to perfect, low-cost illegal copying. In addition, and unlike some previous generation information goods, such as literature, the commercial life-cycle of software may be inherently limited by the rapid evolution of information technology, especially the continuous advancements in computer hardware, which may render today's valuable software much less valuable tomorrow, since software requires complementary hardware in order to function and add value. This further increases the risks for the software creator.

150. Generally speaking, in addition to copy protection, software creators expect that they will retain the decision rights over how their software will evolve. And this is an outcome that benefits the market, as having a single locus of control means that careful thought will go into creating a design that will attempt to maximize

the value of the software to the greatest amount of use. In contrast, an example of what can happen when software products devolve from a single design and fragment into a variety of not 100% compatible products can be seen with the history of the UNIX-style operating system.[82] Since its initial development at Bell Labs, many computer scientists and engineers adopted one of the various versions of the UNIX operating system. However, after Bell Labs gave up control over UNIX, and the source code became freely available, and as the popularity of UNIX-based workstations increased in the market during the 1980s, multiple, incompatible versions of the UNIX operating system standard were introduced to the market by workstation manufacturers that modified the UNIX code to differentiate their products (e.g., Solaris from SUN, AIX from IBM, HP/UX from HP, and Ultrix from DEC). Although similar, the software written to run on one workstation platform often had to be modified to run on another manufacturer's workstation. This market outcome, termed by some the "UNIX Wars," had a number of negative outcomes, including 1) confusion for users and buyers, 2) reduction in the likely availability of complementary application software, and 3) significantly increased learning costs. This confusion, and the resulting technical skills required to use UNIX, greatly limited its success in personal computer markets and, in particular, its appeal to the overwhelming majority of consumers using personal computers.[83]

C. History and Context Regarding Legal and Practical Control of APIs

151. There is a long-established history of companies exerting proprietary control over pre-written programs that developers use to develop other programs—i.e. lines of code and programs that developers invoke to make programs, including APIs. Indeed, there is a commercial market of companies who provide and commercialize such software, exerting both legal and practical control over them, yet, at the same time, distributing them widely so that developers become familiar with them and they become popular. Complicating the broader picture is the fact that software products, as innovative and dynamic as they are, are also especially at risk. A 1995 publication on the history of the software industry in the United States explains:

> "[An] important distinction is between software and the production of other economic commodities. On the one hand, software, like other commodities, requires inputs that have alternative uses and, once produced, has economic value as an intermediate or final good. But, software is also an unusual economic commodity because its marginal costs of reproduction are very low or negligible. The low costs of software reproduction imply that society must grant businesses some right to control reproduction (and charge higher prices

[82] *See, e.g.*, Salus, P., *A Quarter Century of UNIX*, Addison-Wesley, 1994

[83] Notably, Google itself cited the negative outcomes of the UNIX Wars as support for its position that it should be able to exert control over Android compatibility. In a 2009 exchange between Alan Eustace, Google's SVP of Knowledge, and Andy Rubin, regarding a draft communication to Intel regarding violation of the Android anti-fragmentation agreement, Eustace wrote that Google would not support Intel's "Android compatability [sic] hack for cell phones, and I am categorically opposed to it. Intel signed a non-fragmentation agreement as part of their joining the Open Handset Alliance. You are free to pitch [Intel's non-compatible version of Android] to these companies, but not with the promise of compatability [sic] or access to the Android marketplace. Unix was held back for 20 years because every company tried to make a better/incompatible version. We fragmented the market, and allowed Windows to take over the world. We have a chance to not repeat this fundamental mistake on cell phones." (Trial Exhibit 181)

> than the cost of reproduction) if investments are to be made in software creation, especially packaged software."[84]

152. Given the unique vulnerabilities of original works in this segment of the market, there is no misunderstanding that the companies who have invested in creating APIs or similar code have every right to monetize and control their use, even if that material is distributed for free in some contexts, or becomes well-known and popular. Developers and users understand that providers of APIs and similar code, in particular pre-written programs such as the 37 Java APIs at issue, expect to control use of those programs and/or expect to monetize those programs.

153. In fact, there always has been, and continues to be, robust debate about the degree and nature of control to exert in business models governing APIs. An IBM Systems column emphasizes the range of model choice considerations:

> "Companies have had APIs at their disposal for a long time. Existing interfaces to systems have been exposed for internal developers and partners to access. However, each time one of these APIs is created, it is done with a unique effort. Setting up the API, making it available to the audience and applying security are all implemented on a unique basis. There is no common methodology or management around these interfaces."[85]

154. The prevalence of this range of possibilities in the software industry indicates that, contrary to Dr. Astrachan's and Dr. Cattell's assertions, there is no singular, monolithic understanding by developers that APIs are free for the taking and totally unconstrained. Rather, the opposite is true. Participants in the software industry have debated the relative merits of different business approaches, but everyone has always understood that companies, such as Sun and Oracle in this case, may choose to hold their API code as proprietary, exert control over the use, copying and distribution of that code, and directly monetize that code.

155. Software platform distribution has changed over time. A mass-market for distributed personal computing platforms developed in the 1980s and 1990s and there was a corresponding growth in the market for prepackaged, application software.[86] Investment in creating such platforms led to business models which involved control over the ability to create and evolve application programming interfaces (APIs).[87] For example, in 1983, Sytek developed its NetBIOS API as a customized product for IBM, which was implemented

---

[84] Steinmueller, W. E., "'"The US Software Industry: An Analysis and Interpretive History," in *The International Computer Software Industry*, D.C. Mowery (ed.), Oxford University Press, 1995, at p. 4

[85] *See* "The API economy journey map: How are you doing?" IBM Systems, https://www.ibm.com/blogs/systems/the-api-economy-journey-map-how-are-you-doing/ (accessed Feb. 8, 2016)

[86] Mowery, D.C., (Ed.), *The International Computer Software Industry: A Comparative Study of Industry Evolution and Structure*, Oxford University Press, New York, 1996

[87] West, J., J. Dedrick, "Innovation and control in standards architectures: the rise and fall of Japan's PC-98," *Inform. Syst. Res.* 11 (2), 2000, pp. 197–216.

across its distributed PC-LAN in 1984.[88] Microsoft introduced its own distributed computing API in 1987, and by the end of the decade, there were multiple competitive, proprietary APIs for LAN solutions.[89]

156. A 2007 paper considering the history of APIs debates choices regarding the appropriate level of control in API strategy, and refers to "[s]oftware licensing and control over APIs" as a "lever of power" in the context of distributed software (such as the API packages in the current case).[90] It has long been understood that platform owners make strategic choices that sometimes involve exerting control over copying and use of APIs, sometimes involve relinquishing control and sometimes involve finding a balance between approaches, to protect their investment. For example, a 2008 paper observes that "Google's terms of service does not allow you to use their APIs with their ads removed. This brings up the issue of API licensing and the fine line between enabling in order to create value while protecting your monetization."[91] (Google's control over its own APIs will be discussed in more detail below.)

157. This is the context in which software industry and developer expectations regarding APIs evolved, and there has been a continuous understanding that copying, reuse or reimplementation of APIs may not be permitted, depending upon the desires of the API owner. In fact, in a 2014 discussion of this issue, industry participants observed: "It's *always* [emphasis added] been a legal grey area whether an API that a company publishes is reusable and I would say it just became more dangerous to use someone's API design without consulting them first."[92] The article concludes that "Best practices and common sense dictate that if Company B wants to use Company A's APIs, it should talk with Company A first."[93]

158. There are many examples, and a well-established industry custom and common practice, where implementing or reusing APIs is decidedly and specifically *not* permitted by the creators, and developers understand that reality.

159. One specific example of a set of APIs which industry participants understood to be proprietary and subject to the owner's control are the Amazon Web Services APIs. One industry observer noted that

---

[88] Fleig, C.P., "Big Blue keeps a close eye on open architecture," *Network World*, Oct. 20, 1986, p. 1; *See also* "NetBIOS - History and Terminology," Liquisearch, http://www.liquisearch.com/netbios/history_and_terminology (accessed Feb. 8, 2016)
[89] Welch , K. P., "Interprogram communications using Windows' dynamic data exchange," *Microsoft Systems Journal*, 2.5 (Nov. 1987), p13; Dortch, M., "Many Paths, One Mountain: PC-Mini Integration," *InfoWorld*, Nov. 27, 1989, p. S12. It is also worth noting that, during the 1984-1990 timeframe, IBM, Microsoft, Novell and others provided differing implementations of LAN APIs that, nevertheless, achieved similar distributed operability for networked OS/2 hardware systems.
[90] O'Reilly, T., "What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software," *Communications & Strategies*, No. 1, p. 17, Q1, 2007, p. 20.
[91] *See* "Open APIs, Mashups and User Innovation," M. Weiss, http://timreview.ca/article/168 (accessed Feb. 8, 2016)
[92] *See* "Oracle v. Google ruling shows why cloud players may have steered clear of Amazon APIs," B. Darrow, https://gigaom.com/2014/05/09/oracle-v-google-ruling-shows-why-cloud-players-may-have-steered-clear-of-amazon-apis/(accessed Feb. 8, 2016) quoting Steve Wilmott, CEO of 3Scale, an API management company
[93] *Ibid.*

"many cloud providers hesitated to support Amazon Web Services APIs from the get-go," due to Amazon's treating of these APIs as proprietary and the legal uncertainty around them.[94] For example, it was noted that one company, Eucalyptus, added the Amazon Web Services APIs to its own offerings and did so subject to a license.[95] By contrast, another company, Rackspace, declined to use the APIs or take a license to them, and it was understood in the industry that this was to avoid potential litigation that might result from such use.[96]

160. Apple, likewise, places limitations on its licensing of its XCode SDKs and APIs for creating iOS applications. For example, the "Xcode and Apple SDKs Agreement," asserts intellectual property rights in the XCode APIs and imposes limitations on their copying, use and distribution.[97] First, the agreement defines "Apple SDKs" as "the Mac SDK, and the Apple-proprietary Software Development Kits (SDKs) provided hereunder, including, but not limited to, header files, APIs, libraries, simulators, and software (source code and object code) labeled as part of the iOS SDK, watchOS SDK and/or tvOS SDK and included in the Xcode Developer Tools package for purposes of targeting Apple-branded products running iOS, watchOS, or tvOS."[98] Thereafter, Apple imposes limitations on the use of the Apple SDKs, including the APIs. For example, Apple only permits use of its APIs to test and develop applications specifically for use with Apple-branded products, and prohibits distribution of applications using its APIs absent entering into a separate written agreement with Apple.[99] Indeed, early in the history of the iPhone, Apple even required developers to sign an NDA to use the iPhone SDK.[100]

161. Similarly, Microsoft has always closely controlled its APIs and that has been an enormous source of Microsoft's proprietary advantage.[101] For example, Microsoft has always provided its Windows APIs subject to restrictions. In 1998 one commentator observed:

> "Microsoft's most powerful competitive advantage has been the ownership of the Windows operating system, which ships on nine out of every 10 personal computers in the world. Microsoft has become one of the most profitable companies in the world because it owns the underlying technology that drives PCs. Through its ownership of the operating system, it controls the critical device drivers (software that connects the hardware and the software) as well as the critical APIs, or application programming interfaces (software that connects

---

[94] *Ibid.*
[95] *Ibid.*
[96] *Ibid.*
[97] *See* Xcode and Apple SDKs Agreement, Apple Inc., https://www.apple.com/legal/sla/docs/xcode.pdf (accessed Feb. 8, 2016)
[98] *Ibid.*, at ¶ 1
[99] *Ibid.*, at ¶ 2.2 A
[100] *See* Apple Drops NDA for Released iPhone Software, B. Wilson, http://www.cnet.com/news/apple-drops-nda-for-released-iphone-software/ (accessed Feb. 8, 2016)
[101] Iansiti, Marco, and Roy Levien, *The Keystone Advantage: What the New Dynamics of Business Ecosystems Mean for Strategy, Innovation, and Sustainability*, Boston: Harvard Business School Press, 2004, p. 88

> applications to the operating system). No one can copy Windows APIs or device drivers, and Microsoft can change them, eliminate them, or upgrade them, whenever it sees fit."[102]

162. There were clear benefits to developers, consumers and Microsoft from this control over implementation of its APIs. As one observer put it: "Win32 APIs were strategic for Microsoft and generated huge backward compatibility efforts so that no applications 'broke' as Windows upgrades were released."[103] Another commentator explained that Microsoft's control over its APIs to ensure familiarity to developers and end-users assured compatibility and stability:

> "Microsoft has provided a crucial degree of stability in the software industry by ensuring that its application programming interfaces (APIs) remained consistent across different generations of technology. Application developers write programs that call on various APIs to perform routine functions, which greatly reduces the cost of writing software programs. It is important to developers that an operating system have a consistent set of APIs because it ensures that programs that work on one version of an operating system will also work on other versions. Software developers can also be confident that their software applications will not break when new versions of the operating system are released. This, in turn, ensures a familiar experience for developers (and ultimately for end users) and reduces learning costs."[104]

163. Microsoft's long standing strategy has continued with its more recent services as well. For example, Microsoft's Bing Search API is subject to restrictions that only permit copying and use of the APIs in the context of particular end-user facing websites or applications, and Microsoft charges for use of the APIs. Microsoft expressly retains "all right, title, and interest in and to the Services (including the API and Bing results) and all intellectual property rights in any of these."[105]

164. There are numerous other examples of APIs that are controlled and proprietary, even when they are popularized among industry participants. For example, the Unreal engine SDK, a set of software libraries and APIs provided within a software development kit, are used and invoked by many videogame developers.[106] Epic, the company that offers the Unreal engine, gives the SDK away for free so that the tools become popular, but when a user of the SDK makes money on a game built using the tools, he must pay a royalty to Epic.[107]

---

[102] Cusumano, Michael A. and David B. Yoffie, *Competing on Internet Time: Lessons from Netscape and its Battle with Microsoft*, New York: The Free Press, 1998, p. 147.

[103] Transforming Global Information and Communication Markets, Information Revolution and Global Politics Series, MIT Press, 2012 at p. 282.

[104] Iansiti, Marco, and Roy Levien. *The Keystone Advantage: What the New Dynamics of Business Ecosystems Mean for Strategy, Innovation, and Sustainability*. Boston: Harvard Business School Press, 2004, p. 88

[105] *See* "Bing Search API," Microsoft Azure Marketplace, http://datamarket.azure.com/dataset/bing/search#terms at 1; "FAQ for the Bing Search API: Windows Azure Marketplace," Microsoft Corp., https://onedrive.live.com/view.aspx?resid=9C9479871FBFA822!110&app=Word&authkey=!AInKSlZ6KEzFE8k (accessed Feb. 8, 2016)

[106] UE4 Libraries You Should Know About, B. Bramer, Unreal Engine, https://www.unrealengine.com/blog/ue4-libraries-you-should-know-about (accessed Feb. 8, 2016)

[107] If You Love Something, Set It Free, T. Sweeney, Unreal Engine, https://www.unrealengine.com/blog/ue4-is-free; https://www.unrealengine.com/eula (accessed Feb. 8, 2016); *see also* "Unreal® Engine End User License Agreement," Unreal Engine, https://www.unrealengine.com/eula (accessed Feb. 8, 2016)

Even mobile platform providers beyond Sun, Oracle and Google, such as RIM, have always controlled their APIs.[108] Similarly, a 2009 article observed that "mobile operator may choose to differentiate its platform by offering proprietary APIs."[109]

165. More recently, the debate has taken the form of whether it is a better model to permit completely unrestricted access to APIs, retain APIs as wholly confidential, or take a middle ground approach where APIs are distributed subject to restrictions. For example, the Public Broadcast System (PBS) controls use of its COVE (Comprehensive Online Video Ecosystem) API, designed to manage the multitude of video formats and viewing devices people use, and the Localization API which handles highly complex data, to determine what PBS content and stations were available in specific locations. Different levels of control and considerations for distributions rights are defined for Public, Partner, and Private APIs (Public: "distribution rights are not an issue," Partner: "limited distribution rights can be shared with trusted partners," Private: "limited distribution rights")[110]

166. Similarly, observers debate the relative merits of completely "open" or "closed" models, or a middle strategy of "managed" APIs where copying and use are limited by restrictions. As one technology journalist explains, "For CIOs, deciding whether or not an API is open or closed should be a key question as you develop the overall API business strategy."[111] The article continues that in some circumstances a "managed API" approach, where the API is subject to restrictions, may be appropriate, and "[t]his allows you to retain control over the API."[112]

167. Making these choices is not only widely debated, but a complex matter for a number of reasons. Even for businesses that actively seek to transition from tightly controlled product-centered development to more open, platform-centered engagement, there are important tradeoffs to consider, as "the freedom of changing APIs, user interfaces, data models, etc. is significantly constrained by the platform strategy."[113] Furthermore, there is extensive variance in choices over how far an API ecosystem can be opened, based on the business in question. In industries such as healthcare or finance, for instance, APIs must be incorporated into business

---

[108] A Developer's-Eye View of Smartphone Platforms, P. Wayner, *InfoWorld*, http://www.infoworld.com/article/2675582/application-development/a-developer-s-eye-view-of-smartphone-platforms.html?page=2 (accessed Feb. 8, 2016) ("'We created our own proprietary APIs,' explains Mike Kirkup, a manager of developer relations at RIM.")

[109] Goncalves, V. and P. Ballon, "An exploratory analysis of Software as a Service and Platform as a Service models for mobile operators," in "Intelligence in Next Generation Networks," *ICIN 2009*, vol., no., pp.1-4, 26-29, Oct. 2009

[110] *See* Speaker Deck, APIStrat, https://speakerdeck.com/apistrat for detailed information on the topics covered at APIStrat 2015, which was held in Austin, TX from Nov. 18-20, 2015.

[111] *See* "Open or Closed API: Six Guidelines to Help CIOs Make the Call," L. Lawson, Programmable Web, http://www.programmableweb.com/news/open-or-closed-api-six-guidelines-to-help-cios-make-call/2012/08/29 (accessed Feb. 8, 2016)

[112] *Ibid.*

[113] Bosch, J, "From software product lines to software ecosystems" in *Proceedings of the 13th International Software Product Line Conference*, pp. 111-119, August, 2009, Carnegie Mellon University

practices with utmost attention to regulatory requirements, the proper handling of highly sensitive personal information and the basic privacy expectations of clients.[114] However, other industries not saddled by statutory compliance imperatives may instead place emphasis on the benefits stemming from broadened access to their portfolio of APIs. As explained in a tutorial specifically geared towards the decision processes governing the choice of open or closed APIs as part of a business strategy, open APIs can offer a rapid pathway to achieving developmental goals. The tutorial explains:

> "An open API publisher is usually seeking to leverage the ever-growing community of free-agent app developers. This will allow the organization to stimulate development of innovative apps that add value to its core business, without investing directly in development efforts – it simultaneously increases the production of new ideas and decreases dev costs.
>
> An open API may be used by internal developers but it is fair to say – in most cases – the success of an open API program will depend on its ability to attract external developers and help them create truly valuable new apps that people actually want to use. Open API publishers need to engage developers and they need to make sure these developers are successful."[115]

168. Open APIs, however do not represent the only means by which to interact with developers and reach key business benchmarks. Closed, or private APIs also offer routes to internal efficiency that can generate collateral benefits vis-à-vis developer relations. The same tutorial explains:

> "Private APIs can significantly reduce the development time and resources needed to integrate internal IT systems, build new systems that maximize productivity and create customer-facing apps that extend market reach and add value to existing offerings. Rather than creating siloed applications from scratch, [developers] can draw from a common pool of internal software assets.
>
> Essentially then, the goal of a private API program is to enable internal developers who are building new applications that leverage existing systems. Therefore, the needs and preferences of these [developers] should drive the decisions made by business managers and interface developers who are implementing the program.
>
> …Crucially, it is vital to handle the ongoing management of any API program, to ensure the security and performance of backend systems is maintained over the long term."[116]

169. The tutorial also emphasizes Partner APIs as "a hybrid form of the open and private interface models. This kind of API is usually implemented to support applications built by developers within an organization that

---

[114] *See* An API for Healthcare Providers, True Vault, https://www.truevault.com/healthcare-api.html (accessed Feb. 8, 2016) for examples of the complex considerations undertaken by the creators of healthcare-specific APIs
[115] *See* API Strategy 201: Private APIs vs. Open APIs, API Academy, http://www.apiacademy.co/resources/api-strategy-lesson-201-private-apis-vs-open-apis/ (accessed Feb. 8, 2016)
[116] *Ibid.*

has an existing relationship with the API publisher."[117] In other words, APIs may be made available with select parties, subject to restrictions, through contracts or otherwise.

170. As the above quote regarding private APIs illustrates, APIs that are subject to control offer advantages with respect to both cooperative development as well as operational integrity perspectives. API "evangelist" Chris Haddad, argues similarly that "governance keeps the environment from devolving into chaos by managing people, policies and processes. API management systems enforce operational controls on essential development lifecycle functions and runtime technologies."[118] The risks of API mismanagement are very real, and can impact business operations in any industry.

D. The API Economy: Legal and Practical Control of APIs and Other Software Development Tools Encourages Creativity and Investment by Companies like Sun and Oracle

171. Oracle is not alone is seeking to monetize its APIs and seeking to control the copying and use of its APIs in order to do so. Many companies incorporate such control into their business strategies. Control of APIs and other software development tools is a known and expected dynamic in the dissemination of software, and encourages innovation and investment by companies like Sun and Oracle to create new software. This established and widespread understanding is particularly important as APIs become more prevalent across software contexts, as part of what is sometimes referred to as the "API economy." Given the increased importance and widespread creation of APIs, there is an increased need to ensure that investment in API design is protected. Creators must be rewarded for their creative activities, so that they have incentives to continue such creative labors and bring further value to the API economy.

1) <u>The Creative, Expressive Value of API Packages is Evident in the Emergence of the API Economy</u>

172. The prominence, importance and trajectory of API package design has become so clear in recent years that the term "API economy" is now associated with the rapidly expanding universe of creative expression, adoption and value that touches virtually every sphere of commercial activity.[119] A 2015 Deloitte report on critical business technology trends expands on the deeper value of their cohesive creative properties:

> "Application programming interfaces (APIs) have been elevated from a development technique to a business model driver and boardroom consideration. An organization's core assets can be reused, shared, and monetized through APIs that can extend the reach of existing

[117] *Ibid.*
[118] *See* "Five Actions for Maximizing Your API Value," C. Haddad, IT Briefcase, http://www.itbriefcase.net/five-actions-for-maximizing-your-api-value (accessed Feb. 8, 2016)
[119] Medrano, R., "Welcome to the API Economy," *Forbes*, Aug 29, 2012, *available at*: http://www.forbes.com/sites/ciocentral/2012/08/29/welcome-to-the-api-economy/#7fbea7e16d396df08abb6d39 (accessed Feb. 8, 2016) The author writes: "APIs are a key growth driver for hundreds of companies across a wide range of industry sectors."

> services or provide new revenue streams. APIs should be managed like a product—one built on top of a potentially complex technical footprint that includes legacy and third-party systems and data."[120]

173. There is significant evidence that, in recent years, many more companies have invested in the creation of APIs:

> "The API revolution is upon us. Public APIs have doubled in the past 18 months, and more than 10,000 have been published to date. The revolution is also pervasive: Outside of high tech, we have seen a spectrum of industries embrace APIs—from telecommunications and media to finance, travel and tourism, and real estate. And it's not just in the commercial sector. States and nations are making budget, public works, crime, legal, and other agency data and services available through initiatives such as the US Food and Drug Administration's openFDA API program."
>
> "The idea behind APIs has existed since the beginning of computing; however in the last 10 years, they have grown significantly not only in number, but also in sophistication. They are increasingly scalable, monetized, and ubiquitous, with more than 12,000 listed on ProgrammableWeb, which manages a global API directory."[121]

174. There is growing recognition that API packages provide far more than mere technical functionality. Their unique abilities to express creative organization allow them to serve as conduits to all aspects of commercial life:

> "[W]hy is there so much industry energy and investor excitement around APIs? The conversation has expanded from a technical need to a business priority. Jyoti Bansal, founder and CEO of AppDynamics, believes that APIs can help companies innovate faster and lead to new products and new customers. Bansal says, 'APIs started as enablers for things companies wanted to do, but their thinking is now evolving to the next level. APIs themselves are becoming the product or the service companies deliver.' The innovation agenda within and across many sectors is rich with API opportunities. Think of them as indirect digital channels that provide access to IP, assets, goods, and services previously untapped by new business models. And new tools and disciplines for API management have evolved to help realize the potential."[122]

---

[120] Briggs, B., and C. Hodgetts, "Tech Trends 2015: The Fusion of Business and IT," *Supply Chain* 247 (2015), p. 21, *available at*: http://dupress.com/periodical/trends/tech-trends-2015/?id=us:2el:3dc:dup:eng:cons:tt15_(accessed Feb. 8, 2016)

[121] *Ibid.* *See also* Evans, P. and R. Basole, "Revealing the API Ecosystem and Enterprise Strategy via Visual Analytics", *Communications of the ACM*, February 2016, v. 59, n. 2, p. 27 (accessed Feb. 8, 2016) The intricacies and imperatives of the API economy command the attention of virtually every modern industry. *See also* API World Conference, http://apiworld.co/conference/; API Days Conference, http://sf.apidays.io/; I Love APIs Conference, http://iloveapis.com/; API Craft Conference, http://www.apicraft.org/Speaker Deck, APIStrat, https://speakerdeck.com/apistrat (all accessed Feb. 8, 2016) API design has been the subject of some 200+ professional conference talks since 2013—just at one forum, the APIStrat annual conference. Recent APIStrat participants include IBM, Walgreens, Intuit, Uber, Ford, Twitter, Rabobank, WIPRO, GSMA and Google. Talks have delved into the critical value of API package creation across industries as wide-ranging as banking, education, publishing, healthcare, travel, aerospace, retail, television, the public sector and humanitarian non-profit work. Other annual conferences include API:World, ILoveAPIs, API Days, and API-Craft, among others. APIStrat participants form a mosaic of organizations from industry leaders to tech pioneers to public institutions.

[122] Briggs and Hodgetts, p. 23

175. In a similar vein, as Mandy Waite, a Google Cloud Developer Advocate explained at the most recent APIStrat conference:

> "Everything at Google is—or has—an API."[123]

176. Every enterprise is different, and the presence or roles of specific clients, partners and suppliers will vary from industry to industry, and from business to business. Since my first report, an article appearing in the *Communications of the ACM* on the "API Ecosystem" notes that:

> "Firms are finding APIs to be beneficial in a variety of ways. The initial provider can use an API as a way to create new revenue streams, by offering access to already existing digital information through a range of different business model (including subscription, license, freemium, or pay-as-you-go)."[124]

177. In connection with the API economy's diversity of business models and increased generation of APIs as valuable works, there has also been growth in API development tools and emergence of divergent design philosophies and debate about the creative process that drives the API economy.[125] For example, recently there has been debate about whether better API design should prioritize the preferences of developers or the content of source code.[126] This type of debate suggests that creativity is being supported by the current market incentives.

2) The Role of Control in the API Economy

178. It is estimated that approximately 90% of APIs are not public.[127] The reason for this is that different API owners have different needs that are served best by different levels of access. A report by API management technology company 3Scale, explains:

> "Many people assume that API usage involves the opening of a public developer community with access for all, which is not always appropriate for every business. This scenario, however, represents only a small fraction of the APIs in use—many APIs are successfully used solely within the company creating them or with limited sets of close partners."[128]

---

[123] *See* From AdWords to Firebase: The Road to API Nirvana, M. Waite, https://speakerdeck.com/apistrat/from-adwords-to-firebase-the-road-to-api-nirvana, at Slide 2

[124] Evans, P. and R. Basole, p. 27

[125] Louvel, J., "Community Debates API Specification Alternatives," *InfoQ*, Feb. 18, 2015. Available at: http://www.infoq.com/news/2015/02/api-alternatives (accessed Feb 8., 2016); *See also* "What's the best way to write an API spec?" Y-Combinator, https://news.ycombinator.com/item?id=8912897 for more detailed interactions on the specific debates concerning API design tools and documentation procedures (accessed Feb 8. 2016)

[126] Riggins, J., How To Design Great APIs With API-First Design and RAML, ProgrammableWeb, Jul. 10, 2015. Available at: http://www.programmableweb.com/news/how-to-design-great-apis-api-first-design-and-raml/how-to/2015/07/10 (accessed Feb. 8, 2016)

[127] Willmott, S. and G. Balas, "Winning in the API Economy," p. 32, *available at* http://www.3scale.net/wp-content/uploads/2013/10/Winning-in-the-API-Economy-eBook-3scale.pdf (accessed Feb. 8 2016)

[128] *Ibid.*, p. 32-33,

179. The divergent needs of businesses necessitate flexible options for the management of their copyrighted property. Retention of control can provide businesses with an additional justification for the costs and energies associated with the creative development of API packages. Briggs and Hodgetts explain that in between the phases of creation and monetization, issues of access are central. They explain:

> "Technology teams striving for speed and quality are finally investing in an API management backbone—that is, a platform to:
>
> • Create, govern, and deploy APIs: versioning, discoverability, and clarity of scope and purpose
>
> • Secure, monitor, and optimize usage: access control, security policy enforcement, routing, caching, throttling (rate limits and quotas), instrumentation, and analytics
>
> • Market, support, and monetize assets: manage sales, pricing, metering, billing, and key or token provisioning"[129]

180. They continue, explaining that "rolling out a successful API strategy is also a complex endeavor presenting new considerations to take into account" Their considerations include:

> • How should different APIs be exposed and to whom?
>
> • How should those services be secured?
>
> • How should their usage be tracked?
>
> • How are access rights managed?
>
> • How do developers, partners and customers establish their identity and get provisioned for the rights they need for the API?
>
> • How does an organization ensure success for the users of its APIs?[130]

181. Briggs and Hodges are not unique in their emphasis on these features of control. The 3scale report makes similar determinations, identifying several important requirements for success that are listed in the excerpted text, shown below in Figure 30.

[129] Briggs and Hodgetts, p. 23
[130] Willmott and Balas, p. 50

Figure 30: 3Scale's Key Requirements for Success in the API Economy[131]

	Key Requirements
Platform Creation	Onboarding, management and communication with partners and customers, as well as seamless management of access rights.
Distribution	Supporting partner integration, tie-ins to affiliate programs and measuring diverse success metrics.
API-as-a Business	Built in billing and payment services to support a wide range of business models.
Internal Innovation	Private and semi-private API-system rollouts to restrict which audiences have access to which APIs.

182. The notion of controlled APIs are an integral part of the growing economy based on platforms, and it is widely recognized that such controlled, proprietary APIs better allow protection of underlying data and resources, and this in turn promotes investment in platform economies. For example, one industry observer noted that through proprietary APIs, an airline company "was able to exert greater control over one of its key business assets: its own data."[132]

183. There are recent examples of companies closing APIs that were once open, due to changes in business strategy. For example, a 2015 Deloitte study observed that: "As some organizations matured, they revised their API policies to meet evolving business demands. Twitter, for example, shifted its focus from acquiring users to curating and monetizing user experiences. This shift ultimately led to the shuttering of some of its public APIs, as the company aimed to more directly control its content."[133]

184. Further, control of copying and use of APIs may be critical to ensure security of data, services and software implementations that are behind the API code. The 2015 Deloitte study observed:

> "It raises significant questions: Can you protect what is being opened up? Can you trust what's coming in? Can you control what is going out? Integration points can become a company's crown jewels, especially as the API economy takes off and digital becomes central to more business models. Sharing assets will likely strain cyber responses built around the expectation of a bounded, constrained world. New controls and tools will likely be needed to protect unbounded potential use cases while providing end-to-end effectiveness—according to what may be formal commitments in contractual service-level agreements. The technical problems

[131] *Ibid.*, p. 60
[132] *See* "Proprietary APIs: A New Tool in the Age of the Platform," P. Simon, http://www.huffingtonpost.com/phil-simon/proprietary-apis-a-new-to_b_6061722.html (accessed Feb. 8 2016)
[133] Briggs and Hodgetts, p. ##

> are complex but solvable—as long as cyber risk is a foundational consideration when API efforts are launched."[134]

185. To support the API economy there are even service providers who provide solutions for companies to exert control over and manage their APIs. For example, Amazon offers to other companies an "API Gateway" solution which "covers of the management of multiple versions and stages of an API, the management of third party access, the all-important authorization and security factors, and automation of an API into infrastructure as well," including "Identity and Access Management" and "Throttle and monitor requests to protect the backend."[135] In other words, entire service offerings driving the API economy emphasize retaining proprietary control over the use of APIs. As discussed earlier, the software industry has long understood that such control over copying, use, reuse and distribution of APIs is a viable and expected business model. That is even more so the case today when such control over APIs enables generation of remarkable value and incentivizes investment in services that are supported by APIs.

3) API Control Has Positive Technical and Economic Consequences

186. Allowing creators of APIs to control their use, reuse or reimplementation will encourage creators of APIs to invest in the time consuming and elaborate process of creating code structures of this kind, which further serve to protect their investment in underlying resources, whether the underlying resources are software, as in the Java APIs in this case, or other forms of underlying data. To allow free use of APIs would have the potential to destroy the investment made by platform creators in designing the APIs themselves and in investing in the underlying resources, and discourage future such investments.

E. Dr. Astrachan's and Dr. Cattell's Assertions that the Java APIs are Free to Copy and Use Improperly Ignore the History of APIs and the API Economy

1) Dr. Astrachan and Dr. Cattell Reach Overly-Broad Conclusions Regarding the Java APIs and All APIs Based on a Narrow Set of Examples

187. Dr. Astrachan and Dr. Cattell provide examples of non-Java APIs that purportedly have led to expectations that APIs in general, including the Java APIs, are free to copy and use.[136] But, as can be seen from the above, there is also a robust history of APIs and similar development tools that are controlled by their owners, through legal and practical means. In fact, for a significant period of time in the software industry there has been a robust debate regarding the relative merits of controlled, proprietary APIs and development tools, as a business model, versus completely unconstrained business models. This history demonstrates that

---

[134] Briggs and Hodgetts, p. ##

[135] *See* Amazon Introduces API Gateway, Aims to Dominate Yet Another Area, B. Kepes, Network World, http://www.networkworld.com/article/2942602/cloud-computing/amazon-introduces-api-gateway-aims-to-dominate-yet-another-area.html (accessed Feb. 8 2016)

[136] *See* Astrachan Opening Report, ¶¶ 281-305 (discussing particular histories of spreadsheet, Linux and SQL APIs); Cattell Opening Report, ¶¶ 49-51 (discussing particular histories of Linux and SQL APIs)

Dr. Astrachan and Dr. Cattell overreach in drawing sweeping conclusions about APIs in general and attempting to then extend those faulty conclusions to the Java APIs at issue. At most, Dr. Astrachan and Dr. Cattell show that some other APIs come with their own particular, idiosyncratic histories and expectations. The idiosyncratic histories and expectations of these other APIs cited by Dr. Astrachan and Dr. Cattell shed no particular light on the expectations and understanding of the 37 Java APIs at issue here, and for that reason are immaterial. To the contrary, and as elaborated later in this report, the Java APIs at issue have had a long history of control and strong limitations on their copying, use and distribution, and developers in the Java ecosystem have always been able to understand this, and understand that reality to this day.

2) Merely Because APIs are Popular and Known Does Not Mean They Are Free to Copy and Use

188. More generally, Dr. Astrachan and Dr. Cattell assert that the lines of declaring code taken from the Java APIs are popular and well-known, and then extend that proposition to the extreme conclusion that, therefore, they cannot be treated as proprietary.[137] This proposition would undermine the ability of copyright to protect economic incentives to invest in creation at all. It simply cannot be the case that merely because expression – software code or otherwise – has become popular, the creator loses his rights to protect it. Ironically, this point of view would find that copyrighted works that are popular, i.e., have demonstrated a higher market value, would be less deserving of protection than works that have little or no audience. As a practical matter, this cannot be the case and economically it would create serious disincentives for creators, whether Sun or Oracle or any other company, from creating expression and popularizing it.[138]

189. Additionally, Dr. Astrachan points out that Sun and Oracle attempted to popularize the Java platform, and the 37 Java API packages particularly, by educating developers about those prewritten programs and also by working with educational institutions so that students came to be familiar with the prewritten programs that constitute the 37 Java API packages. He cites classes that popularized the software.[139] From there, Dr. Astrachan makes the leap that, therefore, Sun and Oracle gave up all rights to exert control over the Java API packages. This proposition is inconsistent with expected and well-understood software business models.

190. There are numerous examples of companies that distribute their software widely in educational contexts, even for no charge, in order to popularize the software and develop a familiar base of users, so that they are

---

[137] Cattell Opening Report, ¶¶ 35-53; Astrachan Opening Report, ¶¶ 42-95, 171-173, 271-280.
[138] Along these lines, the Court of Appeals stated: "Finally, to the extent Google suggests that it was entitled to copy the Java API packages because they had become the effective industry standard, we are unpersuaded. Google cites no authority for its suggestion that copyrighted works lose protection when they become popular, and we have found none." and that "Google was free to develop its own API packages and to "lobby" programmers to adopt them. Instead, it chose to copy Oracle's declaring code and the SSO to capitalize on the preexisting community of programmers who were accustomed to using the Java API packages." (United States Court of Appeals for the Federal Circuit, *Oracle America Inc. v. Google Inc.* 2013-1021, 1022, p. 52-53)
[139] Astrachan Opening Report, ¶¶ 157-161.

able to charge for the software in other contexts (with the assurance that there will be a large base of users already familiar with the software and willing to pay for it). For example, Microsoft offers training programs that give away Microsoft Office for free to students and teachers.[140] Yet, clearly Microsoft is able to continue to charge for licensed versions of Microsoft Office and has not lost its intellectual property rights in Microsoft Office. Similarly, Autodesk, a major provider of proprietary software and tools, provides teachers and students free versions of its proprietary software, for which it would otherwise charge a license fee.[141]

191. Importantly, this same model is widely adopted with respect to development platforms like the 37 Java APIs at issue. A good example is the Unreal engine SDK, which is provided for free to teachers and students so that the platform become popular, but when a user of the SDK makes money on a game built using the tools, he must pay a royalty to Epic.[142] There is no difference between these promotional activities regarding development tools and other software, and those of Sun and Oracle regarding encouragement of learning and use of the 37 Java API packages in schools.

F. Contrary to Dr. Astrachan's and Dr. Cattell's Arguments, Control Over APIs Ultimately Serves Developers, Platform Owners and Consumers

192. As already established, companies have always differentiated their own products and services through unique API design, and in today's "API economy" it is widely understood that this differentiation drives investment, innovation and competition, and creatives incentives for companies to create the most expressive, appealing and useful APIs. Ultimately developers, platform owners and consumers are all better served by the ability of an API creator to protect and control his own works. Dr. Astrachan's and Dr. Cattell's opinions (Astrachan Report, ¶¶ 271-305; Cattell Report, ¶ 45) are essentially advocating a rule by which companies are no longer permitted to choose the business model they wish to apply to best monetize their APIs. Dr. Astrachan and Dr. Cattell do not attempt to provide any empirical evidence that allowing API creators to control their APIs would have a detrimental impact on the market for APIs or companies who create APIs. Rather, for example, Dr. Cattell suggests (without any support) that control of APIs "might well make the original product less attractive to developers, thus limiting its value" and posits (without any support) that only through unconstrained copying of APIs can the creator extract value. (Cattell Report, ¶ 45) However, the historical range of business models regarding APIs, the extensive history of control of API copying and use as a mechanism for creators to derive value, and the robust API economy that such business models have generated, suggest that Dr. Astrachan's and Dr. Cattell's opinion are not supported by historical evidence.

---

[140] *See* "Students and teachers: You may be eligible to get Office for free!" Miscrosoft Office Blogs, https://blogs.office.com/2014/09/22/students-teachers-may-eligible-get-office-free/ (accessed Feb. 8 2016)
[141] *See* Students, teachers and academic institutions worldwide are eligible for free* access to Autodesk software. Yes, free. We genuinely believe in education. Autodesk, http://www.autodesk.com/education/home (accessed Feb. 8 2016)
[142] *See* "If You Love Something, Set It Free," T. Sweeney, Unreal Engine, https://www.unrealengine.com/blog/ue4-is-free (accessed Feb. 8 2016)

G. Sun has Historically Made it Clear that Use of the Java APIs was Subject to Legal and Practical Restrictions, and the Developer Community Understood This

193. Dr. Cattell and Dr. Astrachan argue that APIs, in general, are purportedly not subject to any constraints on use or distribution.[143] As discussed above, there is extensive evidence that this is simply not the case in the context of APIs and software development tools in general. But, Dr. Astrachan's and Dr. Cattell's assertions in this regard are also inappropriate in light of Sun's and Oracle's actions. The Java community's response and understandings demonstrate that Sun and Oracle have always exerted control over the Java APIs to protect their investment and that the community understood that. The fact that the Java community has always understood that the Java APIs are not free to use without restriction refutes both Dr. Astrachan's and Dr. Cattell's assertions.[144]

1) The "Write Once, Run Anywhere" Proposition of the Java Platform Inherently Involves Control Over the Java APIs and Limitations on Copying, Using or Distributing the Java APIs.

194. The central thesis of the Java platform in general is "Write Once. Run Anywhere." and that has been the case since Java was first introduced into the market, as described in my earlier report. In order to actually achieve this proposition, as a technical matter, control over the copying, use and distribution of the Java APIs has always been necessary. For example, to achieve consistency, compatibility and a coherent Java ecosystem, in which the central tenet of portability across different platforms can occur, Sun and Oracle had to have control over how the APIs could be used. For example, license prohibitions on copying only portions, subsetting, supersetting, or uses that did not pass the compatibility tests provided by Sun and Oracle all acted to achieve the central goal of Java. And each of these limitations on copying, use and distribution of the Java APIs was well-understood by the Java community. The community understood the central goal of the platform and what it took to achieve the goal. For this reason, it is not reasonable for Dr. Astrachan or Dr. Cattell to make sweeping statements about Java APIs or APIs in general, that there is purportedly an expectation that there are no limitations on use of such APIs. Their assertions that no one in the Java community expected Sun or Oracle to exert control over the declaring code and organization of the Java APIs is inconsistent with the basic "Write Once. Run Anywhere." notion of the Java platform itself, which inherently has always required limitations on copying and use of APIs.

---

[143] Cattell Opening Report, ¶¶ 35-53; Astrachan Opening Report, ¶¶ 52-63, 72-78, 157-173, 271-280.

[144] My conclusions in this regard are also supported by the observations of Dr. Schmidt in his report, regarding the understanding in the Java community that Sun and Oracle exerted control over the Java APIs. (Schmidt Report, §VII-J) I incorporate and rely upon those opinions as well.

2) Sun's and Oracle's Licensing Framework has been Clear that Use of the Java APIs was Subject to their Control, and the Community has Expected and Understood these Restrictions

195. A clear indicator that the Java community has understood that the Java APIs were and are subject to legal and practical controls is the fact that Sun has historically offered only restrictive licenses that impose various forms of control over the use and distribution of the Java APIs. The three forms of licenses provided by Sun and Oracle are: (1) commercial license, (2) specification license and (3) GPLv2 with Classpath Exception. The mode of control in each of these scenarios is explained below.

196. **Commercial License.** Sun and Oracle have a history of offering commercial licenses. The key control features in these licenses are twofold. First, the licensees must pay license fees. Second, the license fees that licensees pay are for a license to use the Java APIs *only* in a purely and completely compatible platform. This both increases the scope of the Java platform and ecosystem, and ensures that there is interoperability among implementations of Java and portability—consistent with Java's "Write Once. Run Anywhere" principle. For example, the commercial license agreement between Sun and Danger Inc.—Andy Rubin's mobile platform company prior to Android—required Mr. Rubin and his company to adhere to copying and use restrictions, not to change the Java APIs, to pass the TCK tests, and to pay license fees.[145] Google's CEO, Larry Page, testified he was "aware that many companies take licenses from Sun and now Oracle to use Java."[146] He also could not name "a single company that uses Java APIs that has not taken a license from Sun or Oracle, except for Google."[147]

197. The industry was well aware of these terms and limitations on the APIs through commercial license terms. Each developer contributing to Java technology through the Java Specification Requests in the Java Community Process signs an agreement recognizing Sun's requirements of control.[148] Java developers

---

[145] OAGOOGLE0100036648-OAGOOGLE0100036679 at 677 (Section 2.3: "Compatibility Testing. Successful compatibility testing must be completed by You, or at Original Contributor's option, a third party designated by Original Contributor, to conduct such tests, in accordance with the User's Guide, and using the most current version of the CLDC and M1DP TCKs that were available from Original Contributor one year prior to: (i) Your Internal Deployment Use; and (ii) each release of Compliant Covered Code by You for Commercial Use. In the event that a new release of Compliant Covered Code contains only Minor Changes from the most recent release of Compliant Covered Code, You may use (and must pass) the same version of the TCK that was required to be used with such previous release. "Minor Changes" means (a) bug fixes or modifications meant only to correct errors or defects and/or (b) modifications or additions made to implement "onscreen" graphics, artwork or branding changes; provided that in the case of (a) or (b) such changes do not add new functionality, features or APIs, and do not otherwise materially alter the release of Compliant Covered Code. In the event that You elect to use a version of Upgraded Code that is newer than that which is required under this Section 2.3, then You agree to pass the version of the TCK that corresponds to such newer version of Upgraded Code.")

[146] Page Trial Tr. 487:10-12

[147] Page Trial Tr. 488:2-488:7

[148] OAGOOGLE0100031837 (under the JSPA agreement, developer grants Sun intellectual property rights and allows Sun to commercially license the "RI and TCK together, or the TCK separately"); https://jcp.org/en/participation/members/ (lists hundreds of "JCP Members, comprised of companies, organizations and individuals")

understood the Java commercial licensing scheme, as well.[149] Similarly, a long list of companies that wanted to create technology using Sun's source code and the Java APIs obtained a commercial license to do so.[150]What's more, Google's expert on open source licensing, Chris DiBona, warned Google's employees that the Sun Community Source License was not open.[151] And another Google engineer, Daniel Berlin, advised that "[w]ithout a signed commercial use addendum [to the SCSL], you actually can't use [Sun source code] at all . . . (you'd never be able to deploy it, even internally)."[152]

198. Android's leadership was also familiar with Sun's commercial licensing terms. Andy Rubin, while at Danger and Android Inc., negotiated with Sun for a Java commercial license.[153] After joining Google, he recognized that "Java.lang apis are copyrighted. And [that] sun gets to say who they license the tck to . . . ."[154] Google and Sun went on to negotiate from August 2005 to July 2006, a period during which Google considered taking a commercial license for Sun's source code.[155] Others in Google's management and on the Android engineering team were aware of the licensing requirements, as well.[156] And this remained true even in 2010.[157]

199. **Specification License.** Sun and Oracle have a history of offering the Specification License which, like the commercial licenses, provides very specific control over the Java platform. The Java Specification License states that licensees may copy and use the declaring code and structure, sequence and organization of the Java API packages, as long as the licensee *strictly* creates only a compatible implementation using those APIs and passes the Java compatibility tests, consistent with Java's "Write Once. Run Anywhere." principle. In relevant part, the license states:

> Sun also grants you a perpetual, non-exclusive, worldwide, fully paid-up, royalty free, limited license (without the right to sublicense) under any applicable copyrights or patent rights it may have in the Specification to create and/or distribute an Independent Implementation of the Specification that: (i) fully implements the Spec(s) including all its required interfaces and functionality; (ii) does not modify, subset, superset or otherwise extend the Licensor Name Space, or include any public or protected packages, classes, Java interfaces, fields or methods within the Licensor Name Space other than those required/authorized by the Specification or

---

[149] OAGOOGLE2001355659 (Stephen Colebourne listing the commercial license and OpenJDK license as different ways to get Java); GOOGLE-14-00002326 (Bruno F. Souza, a Brazilian Java developer, explaining the commercial aspects of the Sun Source Community Source License (SCSL) to an open source community); OAGOOGLE0000494636 (A developer wanted to "use Java as a layer between the user and the OS. But because Java is under a commercial license" he could not do it for free.)

[150] OAGOOGLE2009765998 (Attachment D: Commercial Use License to the Sun Community Source License with Samsung); OAGOOGLE0101775461 (Sun Community Source License with Motorola); OAGOOGLE0025475455 (Checklist response to Attachment D: Commercial Use License to the Sun Community Source License with RIM); OAGOOGLE0029214822 (Technology License Distribution Agreement with IBM)

[151] GOOGLE-14-00003518

[152] GOOGLE-14-00039843

[153] Gupta Deposition Tr. 102:4-23 (Danger negotiation); Cizek Deposition Tr. 76:3-11 (Android Inc. negotiation).

[154] TX 18

[155] PX 7 (discussing requirements of "Sun Community Source License").

[156] TX 1 ("Must take a license from Sun"); TX 12 ("either a) we'll partner with Sun as contemplated in our recent discussions or b) we'll take a license")

[157] TX 10 ("we need to negotiate a license for Java")

> Specifications being implemented; and (iii) passes the TCK (including satisfying the requirements of the applicable TCK Users Guide) for such Specification. The foregoing license is expressly conditioned on your not acting outside its scope. No license is granted hereunder for any other purpose.[158]

200. There should not have been any confusion that Sun and Oracle expected control through the use of their pre-written API code under the Specification License. The industry was well-aware of these terms and limitations on the APIs through the Specification License.[159] Java industry reporters have for years recognized that developers implementing Java specifications must maintain compatibility.[160] Google's engineers working on Android understood that using Sun's specification comes with a compatibility mandate.[161] Andy Rubin, the head of Android, interpreted the specification license in the same way.[162] Although Google's lead engineer for the class libraries was not fond of Sun's tight control of its APIs through the specification license, he recognized that it was "the unfortunate reality."[163] Indeed, one engineer went as far as admitting that because Harmony Class Libraries were built using the specification and were incompatible with Sun's specification, Harmony was "doing illegal things [and] (in fact, Android using Harmony code [was] illegal as well)."[164]

201. **GPLv2 with Classpath Exception License.** Most recently (May 8, 2007) Sun and Oracle have offered the GPLv2 with Classpath Exception License, for versions of the Java API packages in the OpenJDK project. This license imposes very significant additional obligations relating to the copying, use and distribution of the Java API packages (including the 37 packages that Google copied). In particular, depending on the technical

---

[158] *See* Java 2 Platform Standard Edition Development Kit 5.0 Specification, Oracle, http://docs.oracle.com/javase/1.5.0/docs/relnotes/license.html

[159] OAGOOGLE0102318356 (Comments from Java community members: "TCK licenses must not be used to discriminate against or restrict compatible implementations of Java specifications"; "I am not complaining about the need to be compatible"; "I understand why you want to keep control and keep everything compatible")

[160] *See* Sun, Microsoft Settle Java Lawsuit, J. Niccolai, *Java World*, http://www.javaworld.com/article/2074908/sun--microsoft-settle-java-lawsuit.html (accessed Feb. 8, 2016) (2001: "all [Microsoft's] future versions of those products must pass Sun's Java compatibility tests, Sun said in a separate statement"); Sun Liberates JDK, Delivers on Open-Source Java Promise, R. Paul, *Ars Technica*, http://arstechnica.com/uncategorized/2007/05/sun-liberates-jdk-delivers-on-open-source-java-promise/(accessed Feb. 8, 2016) (2007: Sun "frequently cited compatibility concerns as the primary reason for perpetuating Java's proprietary status. Sun was concerned that the ability to modify and redistribute Java components would lead to fragmentation."); Apache Foundation to Vote Down Java 7, R.Paul, *Ars Technica*, http://arstechnica.com/information-technology/2010/11/apache-foundation-to-vote-down-java-7-protesting-oracle-abuses/ (accessed Feb. 8, 2016) (2010: "TCK—which is needed to demonstrate a Java implementation's conformance with the language standard—is encumbered by field-of-use restrictions that are fundamentally hostile to open source Java implementors")

[161] GOOGLE-13-00171678 (specification agreement "requires you to use a spec license that only grants IP to people who are compliant with the spec.")(specification licensees must be able to "control [downstream users] who say they are compliant with the spec")

[162] TX 18 ("I don't see how you can open java without sun, since they own the brand and ip.") ("Java.lang apis are copyrighted. And Sun gets to say who they license the TCK to.")

[163] Lee Trial Tr., 1209:17-20 (Furthermore, Apache Harmony members understood how Sun's specification license worked); ASF-GOOGLE-00002863 ("License to get spec -> must implement a tck compliant implementation"); ASF-GOOGLE-00003030 ("The point is that Sun reserves the right to apply or deny the 'stamp of approval' by handing out the TCK."); TX 1047 (Apache FAQ, 2007: "if an entity made a derivative work, or used only part of Harmony's source code in building their implementation, they would still be obligated to obtain their own JCK license for, and test the software themselves")

[164] ASF-GOOGLE-00001393 (Apache's vice president of legal affairs, Sam Ruby, appearing to agree with this statement [Harmony was 'doing illegal things [and] (in fact, Android using Harmony code [was] illegal as well)'])

facts of a particular use of the Java API packages, this license may require companies using code under this license to publicly disclose source code that they would otherwise keep secret or closed-source. As discussed in my initial report, the industry was well-aware of these terms and limitations on the APIs through the GPLv2 with Classpath Exception License. Further evidence of Google's unwillingness to use the GPLv2 with Classpath Exception in Android and industry understandings of the limitations of this license restriction are discussed below. It was well understood that this particular license afforded Sun and Oracle control over the Java APIs.[165]

202. **Other Licensing History.** There are other circumstances that illustrate that Dr. Astrachan and Dr. Cattell are incorrect in their assertions that the Java community purportedly believed or understood that anyone could use the Java APIs in the unlicensed way that Google has done, or that there were no limitations on implementation, copying, use or distribution of the Java APIs.[166]

203. In particular, when the Apache Harmony foundation tried to create an implementation of Java SE, but wanted to attach the Apache license to it, Sun publicly disputed the Apache Software Foundation's legal right to do this.[167]

204. In May 2005, a non-profit research effort was announced to work on an implementation of Java SE 5, called Harmony and developers began working on the project.[168] However, Sun was not willing to grant a license to Harmony to test against the TCK without a "field of use" restriction preventing use of the implementation in any "embedded" contexts (such as mobile phones).[169] The natural reason for this was that Sun wanted to make sure that this non-profit effort, which purported to put the very permissive Apache license on Sun's Java API code, would not become a commercial threat to Sun.[170] Apache made the licensing dispute

---

[165] Also emphasizing the long industry knowledge of Sun's and Oracle's control over the Java APIs, it is notable that before Sun instituted the OpenJDK project, it would not even allow commercial uses of the Java APIs under the GPLv2 with Classpath Exception license. In particular, prior to 2007, there was a non-profit research effort to use the Java APIs in a research implementation, called the "GNU Classpath" project. The project itself posed no commercial risk to Sun or Oracle. However, when parties attempted to use GNU Classpath in a commercial context, Sun took legal action. For example, when a Korean body called WIPI attempted to use GNU Classpath code in a commercial mobile phone effort, Sun immediately engaged, and involved representatives of the U.S. and Korean government to prevent that from happening. Instead, a commercial license was granted to a company called ETRI that facilitated the use of a licensed Java implementation in that Korean mobile phone effort. Once OpenJDK was created, Sun's control over Java obviated the need even for the GNU Classpath research project, and that project ceased. (Smith, Donald (Vol. 01) - 11/20/2015 [2187585] , 237:18-239:9, Nov. 20, 2015); OAGOOGLE0020194367-68 (March 2, 2003 letter to WIPI raising intellectual property issues regarding GNU Classpath); OAGOOGLE0011736496-98 (April 13, 2003 announcement that WIPI ultimately licensed Java)

[166] Cattell Report, ¶¶ 35-53; Astrachan Report, ¶¶ 52-63, 72-78, 157-173, 271-280

[167] My conclusions in this regard are also supported by the observations of Dr. Schmidt in his report, regarding the understanding in the Java community regarding the Apache Harmony dispute, and Sun's and Oracle's exertion of control over the Java APIs in the context of that dispute. (Schmidt Report, ¶¶ 205) I incorporate and rely upon those opinions as well.

[168] *See* Harmony! M. Wielaard, GMane.org, http://article.gmane.org/gmane.comp.java.classpath.devel/5521 (accessed Feb. 8, 2016)

[169] TX 917; TX 1047; TX 1045

[170] Donald Smith Deposition Tr., 235:14-236:18 ("Q. And why was Oracle unwilling to give Apache a TCK without a field of use restriction? A. Well, I think it was Sun that was unwilling than Oracle. Because the field of use restriction is where the commercial side of the Java business is monetized. So if you go back to the very beginning of the entire Java business model, back to the '90s at

very public and Apache and Sun exchanged very public communications about this legal dispute.[171] The entire industry knew that Apache Harmony was legally under question by Sun, and that Sun exerted control over the use of the Java APIs. Indeed, some in the industry were dissatisfied with the fact that Sun was exerting control over its APIs.[172] In response to the dispute, Apache resigned from the Java Community Process Executive Committee, and publicly acknowledged Sun's and Oracle's intellectual property rights (as the "spec lead") in the Java specification, which contains the declaring code and organization of the 37 Java APIs at issue, stating: "Java specifications are proprietary technology that must be licensed directly from the spec lead under whatever terms the spec lead chooses."[173] After Sun's and Oracle's actions regarding Apache Harmony, the Apache Harmony project was abandoned. The Apache website states: "In November 2011, Apache Harmony was retired. The old Harmony website is available online here for archival purposes only"[174] and that "Apache Harmony is retired at the Apache Software Foundation since Nov 16, 2011."[175]

205. Google itself was aware of this public, legal dispute about the use of the Java APIs in Harmony, and that it could not use the Apache Harmony code. For example:

- Google admitted that "Although Sun eventually offered to open source the TCK for Java SE, Sun included field of use restrictions that limited the circumstances under which Apache Harmony users could use the software that Apache Software Foundation created, such as preventing the TCK from being executed on mobile devices."[176]

- Google's Chief Java Architect, Josh Bloch, testified: "Q. Apache Harmony did not have a TCK license, correct? A. That is correct."[177]

---

the very beginning, the model that was set up is that use by ISVs, so companies like Sun and IBM and Oracle and HP and others, that use on general purpose desktop and servers would be gratis. And so it would allow those ISVs to compete against the Microsoft ecosystem. But Sun, as the steward, needed a way to monetize the ongoing development and support of that platform, and so the business model was that there would be a field of use, and that if you were to run Java in a dedicated purpose or -- well, in a non-general purpose case, that would be royalty bearing. You would pay those royalties, and that would help fund the Java platform. And so the entire ecosystem, the entire source code model with -- going back to the very early days with all of those companies was predicated on that. And so, you know, Apache wanted the ability to have that – they basically wanted that particular restriction lifted, and so that would have been unfair to all the other licensees, and moreover, it would have jeopardized the ongoing funding for the development of the Java platform. And so they ultimately -- they tried to find some common ground, but just were never able to."

[171] TX 917; TX 1047; TX 1045; Georges Saab Deposition Tr., 202:11-202:24, Dec. 22, 2015 ("Q Do you think there would be a problem if some third party took the Apache Harmony work product and put it to use?... THE WITNESS: I think that Sun made it very clear at the time that Apache Harmony was being created that they did not think it was a good thing and that – you know, that -- that they -- they -- they didn't endorse it. … Q Why do you believe that? A Because I think it was -- it was widely known in the industry that there was dispute about this and discussion was going on about it.")

[172] For example, when Apache Software Foundation resigned from the JCP executive committee over the licensing issue, two other members of that same group resigned as well (TX 828, Ex. 1045 at 2.)

[173] *See* https://blogs.apache.org/foundation/entry/the_asf_resigns_from_the (accessed Feb. 8, 2016)

[174] *See* http://attic.apache.org/projects/harmony.html (accessed Feb. 8, 2016)

[175] *See* http://harmony.apache.org/ (accessed Feb. 8, 2016)

[176] Lee Trial Tr., 978:18-25

[177] Bloch Trial Tr., 829:9-10

- Google's former Core Library Lead for Android, Bob Lee, testified: "Q. And Apache concluded that a license was required from the spec lead under whatever terms the spec lead chooses; did they not, sir? A. That's the unfortunate reality, yes."[178]

- Andy Rubin recognized in an October 2005 document that Apache Harmony, even in its disputed state, was limited to "non-mobile areas."[179]

- Bob Lee wrote in May 2008, "Sun puts field-of-use restrictions in the Java SE TCK licenses which prohibit Java SE implementations from running on anything but a desktop or a server. These restrictions prevent Apache Harmony from independently implementing Java SE – …not to mention Android (though that's water under the brid[g]e at this point)."[180]

206. It is my understanding that Google claims to have taken the 37 Java APIs from the Apache "Harmony" project. Google's CEO testified that Google did not acquire any IP rights from Sun through its taking of the Harmony materials under the Apache license: "Q. Now, with respect to the question of the Apache license that you mentioned, you are not asserting that Google has any rights to use any Sun intellectual property as the result of the Apache license; correct, sir? A. That is correct."[181]

H. This dispute was extremely public. Its very existence demonstrates that the industry did not understand or expect that the Java APIs were free to use, unconstrained, or that they were free for others to take without providing benefit to Sun or Oracle. Any of Dr. Astrachan's and Dr. Cattell's assertions to the contrary are inconsistent with the historical facts.Google Exerts Legal and Practical Control Over the Android APIs and other APIs

207. In discussing what the industry norms are with respect to APIs it is instructive to examine how Google controls its own APIs. Overall, its approach can be categorized as one of careful and tight control, in contrast to a hypothetical world where they might allow others to do whatever they might want to do with them.

1) Google's Control Over Android APIs

208. Google controls its Android APIs through anti-fragmentation agreements with OEMs that make Android phones. As Hiroshi Lockheimer, the head of Android, testified: "Q. What are the obligations of the anti-fragmentation agreement that are on the manufacturers? . . . THE WITNESS: One of the obligations . . . is that they won't, in fact, quote, fragment the Android API set."[182]

---

[178] Lee Trial Tr., 1209:17-20
[179] TX 7
[180] TX 405
[181] Schwartz Trial Tr., 1541:3-1541:7
[182] Lockheimer Deposition Tr. 159:20-24, 160:15-24 (Dec. 8, 2015). *See* Lin Deposition Exh. 5098; TX 4002 at 13; Brady Deposition Tr. 17:8-18, 18:22-20:17 (July 21, 2011) (Topic 9); Lin Dep. 65:23-66:3, 67:23-68:5 (Dec. 14, 2015); Lin Dep. 261:17-19, 264:8-12 (Dec. 18, 2015)

209. Specifically, anti-fragmentation agreements require that the OEMs "not take any actions that may cause or result in the fragmentation of Android[,]" "only distribute Products that are either: (i) in the case of hardware, Android Compatible Devices; or (ii) in the case of software distributed solely on Android Compatible Devices[,]" "not distribute a software development kit (SDK) derived from Android or derived from Android Compatible Devices[,]" and "not participate in the creation of, or promote in any way, any third party software development kit (SDK) derived from Android, or derived from Android Compatible Devices . . .."[183] If a potential partner wants to distribute Google applications, use the Android branding and trademarks, or gain early access to the Android source code, it must sign one of these agreements (or an agreement that contains an anti-fragmentation provision), limiting the way it can use the Android APIs.[184]

210. Beyond these agreements to control the APIs, Google also has its own Compatibility Definition Documents and Compatibility Test Suites (which do not conform to Oracle's compatibility tests).[185] Google's corporate representative on dealings with OEMs testified: "[f]or each major [Android] operating system release, [Google has] a Compatibility Definition Document (CDD), which identifies all of the APIs and capabilities that [Google] expect[s] to be implemented, and then there is a corresponding set of tests called the Compatibility Test Suite, which validates that every one of those features has actually been implemented correctly and works."[186] The CDD "enumerates the requirements that must be met in order for mobile phones to be compatible with Android . . . ."[187] With regard to the Android APIs, CDD requires that all "device implementations MUST comply with [the following] requirements":

- "MUST NOT omit any managed APIs, alter API interfaces or signatures, deviate from the documented behavior, or include no-ops, except where specifically allowed";
- "MUST conform" Android API build parameters;
- "MUST NOT make any prohibited modifications to . . . java.*[,] javax.*[,] sun.*[,] android.*[,] and com.android.*" package namespaces.[188]

---

[183] *See* Lin Deposition Exh. 5098; GOOGLE-00395207–GOOGLE-00395248; GOOGLE-03374594–GOOGLE-003374599
[184] *See* Brady Deposition Tr. 18:22-20:17, 40:15-24 (July 21, 2011); Lin Deposition Tr. 67:23-68:5 (Dec. 14, 2015); GOOG-00259405; GOOG-00527347-GOOG-00527349; GOOG-00527350-GOOG-00527356
[185] Ghuloum Deposition Tr., 189:7-189:18 ("So Google has created its own Compatibility Test Suite for Android, correct? A Yes, that's correct. Q And Google does not use the Java TCK, in other words, the Oracle compatibility test in order to test Android? A That's correct. Q Those are different tests; the Oracle TCK test of compatibility is a different test of compatibility than the Android Compatibility Test Suite? A Yes, that's correct.")
[186] Lin Deposition Tr., 66:4-19 (Dec. 14, 2015); *see* GOOG-00259405; Lockheimer Deposition Tr., 155:21-158:18 (Dec. 8, 2015); Morrill Trial Tr. 1014:9-13; Camargo DepositionTr. 39:4-40:13
[187] TX 749 at 3; Morrill Trial Tr. 1011:4-7 (CDD provides "binding requirements")
[188] TX 749 at 4-8.

211. To ensure compliance with the CDD, the manufacturers run Compatibility Test Suites, which test "the APIs in a black box perspective."[189] Google further conducts "[s]ome manual testing for compatibility" and "[m]anual testing of Google apps, Market, and other[]" applications before granting approval.[190] Large phone manufacturers complied with these compatibility requirements, bringing millions of Android phones to the market.[191]

212. Google also tightly controls all Android releases and has some proprietary versions of Android and associated APIs that it releases to selected OEMs ahead of the full release.[192] For example, Google has done this in the past with its Honeycomb release.[193]

213. Google believes that control of its APIs through these mechanisms is essential to its own Android ecosystem.[194] Google uses "[c]arrots" and "stick[s]" to convince potential partners to achieve compatibility.[195] The "[c]arrots" include Google Play (Android Market), Google Mobile Services, revenue-sharing, and the Android trademark and brand.[196] The "stick[s]" are the anti-fragmentation agreements and similar provisions contained in the "[c]arrot[]" agreements.[197]

214. Google also controls the APIs for Google Play, the Android app store. While the APIs for Google Play are nominally open-source, applications for the Android platform must adhere to controlled approval processes in order to be a viable option for a developer.[198] As explained in a 2012 *Forbes* article:

> "Based on the level of competition, executives select API operability standards that directly enable the most advantageous partnerships for their company's goods and services…At Google, for example, free APIs from Google Play Services (e.g., 3D MapsGL maps) are playing a significant role in encouraging developers to build apps for the closed Google Play Ecosystem, driving the development of apps for Google-approved Android devices (vs.

---

[189] Brady Deposition Tr., 88:10-14
[190] Brady Deposition Tr., 108:22-110:8
[191] *See* Morrill Trial Tr. 1017:1-12 (750,000 compatible Android activations happened daily in early 2012); Camargo Deposition Tr., 39:4-47:16 (history of Motorola's compliance with CDD and CTS)
[192] *See* Brady Deposition Tr., 52:12-53:2; Lin Deposition Tr., 205:13-206:8; GOOGLE-53-00319805 (Pre-Release Agreement for Gingerbread); GOOG-00138478 (Pre-Release Agreement for Jelly Bean); GOOG-00475452 (Pre-Release Agreement for Lollipop)
[193] Chu Deposition Tr., 156:14-17 ("I'm aware of a Honeycomb early access source code agreement."); GOOGLE-22-00489760 (Google "working on the open source release of Honeycomb" and offering Vizio "an early start on development"); GOOGLE-22-00489771 (draft Pre-Release Agreement for Honeycomb)
[194] GOOG-00259405 ("Compatibility is the engine that drives Android") ("Compatibility program that we run at Google, is key to ensuring health of the ecosystem"); Lin Deposition Tr., 178:19-23
[195] *See* TX 4002; Morrill Deposition Tr., 134:3-12
[196] *See* Morrill Trial Tr. 1015:12-20; Morrill Deposition Tr., 134:3-12; TX 4002
[197] *See* TX 4002; Brady Deposition Tr., 22:19-23:20, 25:2-9, 99:11-100:7
[198] *See* Berlind, D., "Long Live The Private API," ProgrammableWeb, Feb. 6, 2015 (http://www.programmableweb.com/news/long-live-private-api/analysis/2015/02/06) (accessed Feb. 8, 2016) As a contrasting example to the Google Play model, Walgreens, with its Balance Rewards customer loyalty program, opts to combine *closed* API access with *free* app downloads. The Balance Rewards API is a private API, accessible only to approved developers, but the Balance Rewards app is still freely available to the public.

Android forks from other OEMs). As a result, Android has closed the gap with Apple in share of mobile-app revenues, going from 1:5 two years ago to a current ratio of 1:2.3."[199]

2) **Google's Control Over Other APIs**

a) **Google Charges License Fees and Limits Copying and Use of Its Services APIs**

215. Google controls access to, and pricing of, its own APIs for applications. For example, Google charges license fees for use of its APIs.[200] The following figure, below, reflects some of the controls, limitations and restrictions that Google imposes on use, copying and distribution of its own APIs, and associated monetization models. Clearly, Google understands that APIs are a resource that can be tightly controlled and that it can profit through such control:

Figure 31: Copying and Use of Google APIs is Subject to Google's Control[201]

Google Top 5 APIs	Detailed pricing model	Packaging
Maps API Allows developer to embed maps in to their freely accessible web pages or mobile apps	▪ Subscription based plan with a free tier (2.5k calls) and an enterprise tier (100k calls) ▪ Enterprise tier allows 40x more requests per day, analytics, extra features like a demographics layer and tech support ▪ Enterprise pricing not publicly available	▪ Stand Alone w/ Official Libraries
Analytics API Allows developer to view and manage their Google Analytics data	▪ Free for developers to integrate in to their applications or website ▪ Google reserves the right in its discretion to include advertising in the content returned through the APIs	▪ Stand Alone w/ Official Libraries
Translate API Allows translation of text from one language to another	▪ Translation and language detection: $20 per million characters ▪ Usage limits: default limit is set at 2 million characters per day with the option to increase to 50 million characters per day. It is not clear if you will have to pay to increase	▪ Stand Alone w/ Official Libraries
Earth API API that allows developers to add Earth plug-in objects to their sites	▪ When Google provides a brand feature such as a logo through the service the developer must display it as provided and may not delete or alter it	▪ Stand Alone
Prediction API Allows developer to access a cloud hosted machine learning service that makes it easy to build smart apps	▪ Subscription with free tier for the first six months, 100 predictions per day limit ▪ Paid subscription consists of a $10 per month per project fee. Usage beyond 10,000 predictions or streaming updates is charged at $.50/1,000	▪ Stand Alone w/ Official Libraries

[199] Ranade, P., D. Scannell and B. Stafford, "Ready for APIs? Three steps to unlock the data economy's most promising channel," *Forbes*, Jan. 7, 2014 *available at*: http://www.forbes.com/sites/mckinsey/2014/01/07/ready-for-apis-three-steps-to-unlock-the-data-economys-most-promising-channel/#51a1d71e89e5 (accessed Feb. 8, 2016)

[200] *See* https://developers.google.com/maps/pricing-and-plans/#details (accessed Feb. 8, 2016)

[201] *See* Google Maps APIs Pricing and Plans, Google Developers, https://developers.google.com/maps/pricing-and-plans/?hl=en; *see also* Configuration and Reporting API Limits and Quotas, Google Developers, https://developers.google.com/analytics/devguides/reporting/mcf/v3/limits-quotas; Translate API Pricing, Google Cloud,

216. Google controls use of its services APIs through the contracts that assert its intellectual property rights, including copyrights, and contractual rights in the APIs. For example, the "Google Maps/Google Earth APIs Terms of Service," imposes numerous limitations on use of the Google Maps APIs.[202] It defines the "Service" as the "Google Maps/Google Earth APIs" and requires parties using the APIs to confirm: "You understand and agree that Google and its licensors and their suppliers (as applicable) own all legal right, title, and interest in and to the Service and Content, including any intellectual property rights in the Service and Content (whether those rights are registered or not, and wherever in the world those rights may exist)."[203]

217. Similarly, as early as 2005 Google's "Terms and Conditions for Google Web API Service" and as early as 2006 Google's "Terms and Conditions for Google SOAP Search API Service" made clear that Google protected its Google search APIs by copyright. In particular, in the "Intellectual Property" sections of those agreement, Google required: "You agree not to remove, obscure, or alter Google's copyright notice, trademarks or other proprietary rights, notices affixed to or contained within [Google SOAP search/Google Web] APIs" and that Google owns "all intellectual property rights" over "the APIs developed and provided by Google."[204]

218. Google's Vice President for Technical Infrastructure testified that Google's Web Search API, one of its most fundamental services, is highly controlled and other parties must pay Google to use that API:

> "Q The search API at Google is the API for its most fundamental service that it offers to the public; is that right?... THE WITNESS: Yeah. So there are several search APIs. So the Web Search API is, I believe, what you're referring to. So the Web Search API was the first API that we had because we sold Web Search to partners, and so this was actually a private API. As a regular user or developer, you wouldn't be able to call it because you had to pay us in order to -- to call it. So that definitely was the oldest one. Did that answer your question? … Q All right. So people had to pay you to get access to that API? A Search was syndicated to other companies -- Netscape, AOL, Yahoo! -- and they paid us per query using a -- and they -- they sent us the queries using this API. Q Right. And they had to pay you to get access to the API; right? … THE WITNESS: To the service, yes." (Holzle, Urs - (Vol. I) 11/24/2015, 190:10-191:9, Nov. 24, 2015)

b) Google Charges License Fees and Limits Copying and Use of its AdWords API Declarations

219. As detailed at ¶¶ 209, 227-229 of Dr. Schmidt's Rebuttal Report, as of 2005 and thereafter, and in particular in the period 2005-2008, when Google was initially copying and making unauthorized use of the Sun

---

https://cloud.google.com/translate/v2/pricing; Google Maps/Google Earth APIs Terms of Service, Google Developers, https://developers.google.com/maps/terms?hl=en; and, Prediction API Pricing, Google Cloud, https://cloud.google.com/prediction/#pricing (all accessed Feb. 8, 2016)

[202] *See* https://developers.google.com/maps/terms?hl=en, for example ¶ 9 imposes "License Requirements" which restrict use of "Your Maps API Implementation" in ways the limit the way that it can be publicized or held private, the ways in which fees may be charged for it, and controls the enterprise, mobile and public contexts in which the APIs can be used.

[203] *See* https://developers.google.com/maps/terms?hl=en at ¶¶ 1.1 and 6.

[204] Urs Holzle Deposition Tr., Exh. 5004, 5005

and Oracle Java APIs in Android, Google imposed significant limitations on the copying and use of its own AdWords APIs. Google treated those APIs as its proprietary intellectual property. As early as 2005, in order to use Google AdWords APIs, a software developer was required to accept Google's AdWords API Terms and Conditions and "[i]n exchange for use of and access to the proprietary AdWords API and its specifications you agree to be bound by the terms of these AdWords API terms and conditions."[205]

220. The AdWords agreement defines the "AdWords APIs" generally as "a feature of the Google AdWords program and any account management using the AdWords API is also governed by the AdWords terms and conditions between you and Google."[206] More particularly, the agreement states that using the "AdWords API" means: "(A) the use of the mark-up language described in the AdWords API Specifications to (i) access Google servers through the AdWords API, (ii) send information to AdWords accounts using an AdWords API Client, or (iii) receive information from Google in response to AdWords API calls; and/or (B) distributing or developing an AdWords API Client."[207] This definition clearly encompasses API declarations, as set forth in a specification document, and Google expressly defines: "AdWords API Specifications" as "all information and documentation Google provides specifying or concerning the AdWords API specifications and protocols and any Google-supplied implementations or methods of use of the AdWords API."[208]

221. As detailed at §VII-J of Dr. Schmidt's Rebuttal Report, the AdWords API specification referenced in the agreement covers declarations, brief strings of code instructions that are very similar to the Java and Android APIs at issue.[209] Just as in the current case, developers wishing to use the AdWords API would read and navigate the API specification document, to learn and ultimately use the API code specific to particular functions. When developers use the AdWords APIs, they access and utilize software implementations in Google's server-side AdWords service.

---

[205] AdWords API Beta Terms and Conditions, Google AdWords, http://web.archive.org/web/20050130011554/http://www.google.com/apis/adwords/terms.html at Preamble; also: AdWords API Terms and Conditions, Google AdWords, https://web.archive.org/web/20140704175508/https://developers.google.com/adwords/api/docs/terms (nearly identical terms); AdWords API Terms and Conditions, Google AdWords, https://web.archive.org/web/20150406213931/https://developers.google.com/adwords/ api/docs/terms; and, AdWords API Terms and Conditions, Google AdWords, https://web.archive.org/web/20150922081038/https://support.google.com/adwordspolicy/ answer/6169371(similar terms) (all accessed Feb. 8, 2016)

[206] AdWords API Terms and Conditions, Google AdWords, http://web.archive.org/web/20050130011554/http://www.google.com/apis/adwords/terms.html at Preamble

[207] AdWords API Terms and Conditions, Google AdWords, http://web.archive.org/web/20050130011554/http://www.google.com/apis/adwords/terms.html at I (Definitions)

[208] *Ibid.*

[209] I incorporate and rely on Dr. Schmidt's analysis of the nature of the AdWords API declarations and their similarity to the APIs at issue.

222. Notably, Google expressly asserted its intellectual property rights in this material and made copying and use of the AdWords APIs contingent upon developers meeting certain conditions and restrictions. In asserting its rights over the AdWords APIs, Google's 2005 agreement stated:

> "The AdWords API and the AdWords Specifications are, as applicable, the intellectual property and proprietary information of Google. Your right to use, copy and to retain your copy of the AdWords API and the AdWords API Specifications is contingent on your full compliance with this AdWords API Agreement. If you violate all or part of this AdWords API Agreement, your access to the AdWords API may be suspended or terminated without notice. If you decide to terminate your agreement to all or part of this AdWords API Agreement, you must cease all use of the AdWords API and destroy any copies of the AdWords API Specifications, and if requested by Google, certify to Google such destruction."[210]

223. By 2006, this language became even stronger:

> "The AdWords API, all Developer Tokens and the AdWords Specifications are, as applicable, the intellectual property and proprietary information of Google. Any right to use, copy or to retain a copy of the AdWords API and the AdWords API Specifications is subject to and contingent on your full compliance with this AdWords API Agreement. If you violate any part of this AdWords API Agreement, your access to the AdWords API may be suspended or terminated without notice and you have no right to use the AdWords API. If you wish to terminate all or part of this AdWords API Agreement, you must cease all use of the AdWords API."[211]

224. Google prohibits developers from using the AdWords API in any way other than that permitted by the agreement: "You may use the AdWords API to access Google only in accordance with the terms and conditions of this AdWords API Agreement."[212] Google even prohibits developers from using the AdWords API in a client application that it considers to violate its API Agreement: "You may use the AdWords API and AdWords API Specifications to develop and distribute an AdWords API Client only in accordance with the terms and conditions of this AdWords API Agreement" and "[a]ny AdWords API Client (and its development and distribution) must comply with this AdWords API Agreement."[213] Google requires developers to use only

---

[210] *See* AdWords API Terms and Conditions, Google AdWords, http://web.archive.org/web/20050130011554/http://www.google.com/apis/adwords/terms.html at Preamble and IV (accessed Feb. 8, 2016) (Google indicated that its intellectual property rights included copyright as well ("As between you and Google, Google and its applicable licensors retain all intellectual property rights (including all patent, trademark, copyright, and other proprietary rights) in and to the AdWords API Specifications, all Google websites and all Google services and any derivative works created thereof. All license rights granted herein are not sublicenseable, transferable or assignable unless otherwise stated herein.")

[211] *See* AdWords API Terms and Conditions, Google AdWords, http://web.archive.org/web/20060903142934/http://www.google.com/apis/adwords/terms.html (accessed Feb. 8, 2016)

[212] http://web.archive.org/web/20050130011554/http://www.google.com/apis/adwords/terms.html at II (Adwords API Use) (accessed Feb. 8, 2016)

[213] *Ibid.*

the most recent AdWords API,[214] and prohibits them from using only a subset of the parameters of the APIs in their client applications.[215]

225. Google prohibits developers from copying and using AdWords API input fields and data resulting from use of the APIs in any manner that combined these with third party data: "[t]he AdWords API Client must not co-mingle or associate any AdWords API Data or AdWords API input fields with the content of third parties.[216] Similarly, Google prohibits developers from using any information from AdWords APIs in connection with accounts other than Google AdWords accounts: "Any information collected from an input field used to collect AdWords API Campaign Management Data may be used only to manage and report on AdWords accounts."[217]

226. In 2008, Google was asked about its control of the AdWords APIs and Google responded that it was allowed to control copying and use of its APIs, in order to protect the data resources behind the APIs, because the limitations in its agreements only apply to third-party developers and, in Google's view, third-party developers should not have free access to the data resources behind its APIs.[218] In other words, Google controls use of APIs that are very similar to the APIs at issue, quite possibly in order to prevent third-party developers from using those APIs to create competing services.

227. I conclude from the facts discussed above that through licenses, intellectual property rights and practical limitations, Google limits the ability of developers to copy and use APIs in ways that are very similar to those that Oracle is claiming in this case, and thus recognizes that parties have the right to control APIs. This is completely contrary to the position that Google's experts espouse in this matter, that use and copying of APIs should not be limited. Google clearly believes and asserts, when it comes to its own APIs that APIs can be controlled. Google's own positions demonstrate that there is clearly no general industry understanding or practice that APIs are free for anyone to copy and use as Google itself does not even allow that.

3) Google Uses Legal Enforcement to Control Copying and Use of Its APIs

---

[214] *Ibid.*, at III ("All AdWords API Clients must only use a version of the AdWords API that was the most current version of the AdWords API within the 3 months preceding use (a 'Current AdWords API Version'). Any less-current AdWords API Clients must be updated and must not be used, distributed, supported or maintained.")
[215] *Ibid.* at Preamble and IV. Google indicated that its intellectual property rights included copyright as well ("As between you and Google, Google and its applicable licensors retain all intellectual property rights (including all patent, trademark, copyright, and other proprietary rights) in and to the AdWords API Specifications, all Google websites and all Google services and any derivative works created thereof. All license rights granted herein are not sublicenseable, transferable or assignable unless otherwise stated herein.")
[216] *Ibid.*, at III (AdWords API Client Development and Distribution), and related restrictions at http://web.archive.org/web/20060903142934/http://www.google.com/apis/adwords/terms.html at III.
[217] AdWords API Terms and Conditions, Google AdWords, http://web.archive.org/web/20060903142934/http://www.google.com/apis/adwords/terms.html at III.
[218] See PPC Platform Competition and Google's "May Not Copy" Restriction, B. Edelman, http://www.benedelman.org/news/062708-1.html#dataportability (summarizing rationales for the license restrictions offered by Doug Raymond, product manager for AdWords API) (accessed Feb. 8, 2016)

228. Google legally enforces controls against copying and use of its own APIs, and the public record contains multiple examples of such legal enforcement. For example, in 2015, Google sent a cease and desist letter to an Internet service called Routebuilder, that Google asserted was using the Google Maps APIs in violation of Google's agreement governing those APIs.[219] In 2013, Google sent a cease and desist letter to Microsoft demanding that it stop using the YouTube APIs, and ultimately restricted Microsoft's access to the YouTube API to prevent Microsoft from making a good YouTube app for the Windows phone.[220] Similarly, in 2012, Google sent multiple cease and desist letters to Internet sites using the YouTube APIs.[221] In 2006, Google sent a cease and desist letter to an open-source project called Gaia, demanding that the project stop copying and using the Google Earth APIs and ultimately forced the project to do so.[222]

229. For the reasons stated above, and through my experience, research and work regarding management of software development projects and dissemination of software, it is my opinion that developers, users and the general public expect that software development tools, including application programming interfaces, may be subject to legal and practical control and constraints, and in that way expectations are the same as those regarding software in general. In my opinion, while developers may find the 37 Java API packages very valuable and popular, there is no reasonable expectation in the software industry that they are available for free, unlimited use.

VIII. Fair Use Factor 4: Google and Its Customers and Partners Would Not Accept the GPLv2 With Classpath Exception License, and There is Considerable Business and Technical Risk from Use of Such a License

230. Dr. Astrachan asserts that harm to the market for the Java APIs "would have come about as a result of Sun/Oracle's own actions." In particular, Dr. Astrachan argues that Sun created an open-source version of Java SE in 2007, and describes a statement by Google in 2015 that it intends to use the OpenJDK versions of the 37 Java APIs. Dr. Astrachan then argues that: "Although Google did not take the OpenJDK license until 2015, there is no technical reason it could not have done so in the 2007/2008 timeframe as well."[223] Dr. Astrachan only states that OpenJDK was an option from a "technical" perspective, but fails to address whether

---

[219] Google is Forcing Routebuilder to Shut Down, A.Martin, *Medium*, https://medium.com/hacker-daily/google-maps-is-forcing-routebuilder-to-shutdown-615ce42f413a#.q467avy9j (accessed Feb. 8, 2016)
[220] *See* Google Demands Microsoft Pull YouTube App From Windows Phone, M. Honan, *Wired*, http://www.wired.com/2013/05/google-msft-youtube/; Google blocks Windows Phone YouTube app (again), for 'manufactured' reasons, P. Bright, *Ars Technica*, http://arstechnica.com/gadgets/2013/08/google-blocks-windows-phone-youtube-app-again-for-manufactured-reasons/; The Limits of Google's Openness, Microsoft Corp., http://blogs.microsoft.com/on-the-issues/2013/08/15/the-limits-of-googles-openness/ (all accessed Feb 8, 2016)
[221] *See* Google Threatens To Sue Huge YouTube MP3 Conversion Site, Torrent Freak, https://torrentfreak.com/google-threatens-to-sue-huge-youtube-mp3-conversion-site-120619/ (accessed Feb. 8, 2016); *see also* Google threatens to sue Youtube to MP3 conversion web site, L. Bell, *The Inquirer*, http://www.theinquirer.net/inquirer/news/2185556/google-threatens-sue-youtube-mp3-conversion-web-site (accessed Feb. 8, 2016)
[222] *See* "Gaia Project Agrees To Google Cease and Desist," SlashDot, http://tech.slashdot.org/story/06/11/25/225256/gaia-project-agrees-to-google-cease-and-desist (accessed Feb. 8, 2016)
[223] Astrachan Opening Report, ¶¶ 106-107

the business or legal risk associated with OpenJDK would make OpenJDK unacceptable to Google. It is my opinion, for the reasons that follow, that Google did not use and would not have used the OpenJDK code in approximately 2007, given the risks of the GPLv2 with Classpath Exception license. Therefore, the existence of OpenJDK has no bearing on the impact the Google's use of the 37 Java APIs had on the market for Java. Simply put, Google would not have used that code due to the significant business and legal risk, and there is significant risk from use of that code going forward. Thus factor 4 weighs against a finding of fair use.

231. Google's assertions regarding OpenJDK and potential future use of code from OpenJDK, under the GPLv2 with Classpath Exception license are immaterial. There is significant evidence that Google and its customers and partners would not have accepted the significant business and technical risk of that license, and Google's customers and partners are not likely to accept such risks going forward.

232. Using the Java APIs conveyed significant benefits to Google in terms of the speed of development of Android and its stability. Moreover, Sun's open source license was unacceptable to Google.

233. In my professional background, first as a commercial software development project manager and then my academic research and teaching in software engineering management and related consulting/expert witness work, I have a clear understanding of the motivations and risks involved when parties enter into business contracts relating to software. Software development takes a significant amount of skill, and talented software developers tend to be well-compensated. When organizations enter into contracts for software delivery they seek to maximize their risk-adjusted return on this expensive investment. Since the software artifact (the product of the software development process) is an information good, it is what in the business school we term a 'non-rival' good – essentially this means that if I create an information good (a novel, a movie, a piece of software) it has the property that if I sell it to someone (generally in the form of a license to use it), I still possess it. Contrast this with, for example, a typical manufactured good which, once I sell it to someone I have to go create another one – I don't still have the original. This property means that potentially software goods can be highly lucrative, since if I create software that is popular, I can continue licensing it repeatedly without having to re-develop it each time. However, the initial software development process is typically complex and costly, and therefore risky, since when starting a project there is no guarantee of a successful completion (technical risk) nor a guarantee of a willing market to buy it (business risk). Therefore, organizations who develop software seek to carefully maximize their returns when licensing their software, including, for example, preserving their rights to continue licensing the software into the future.

234. In my opinion, Sun and Oracle have chosen licenses that provide Sun and Oracle control over the Java API packages and ultimately the platform, and preserved their investment and potential revenue streams. This was achieved either through license terms giving Sun and Oracle control over use of the platform, or terms of

non-monetary open source licenses that would require contribution back of code modifications or derivatives. The latter form of license would provide disincentives for some companies that would prefer to be free of such restrictions to take the open source license (and would encourage them to, instead, take a commercial license instead).

235. Google was aware of Sun's open source license for Java, but did not accept that license. A company in Google's position would have incentives that were essentially the opposite of a company in Sun's position. Google would have incentives to bring a mobile platform to market very quickly and to quickly achieve adoption by many handset manufacturers, and would negatively perceive licensing terms that required relinquishing control, or limiting technical options for Google or its OEM customers, all else being equal. Based on my experience, a platform that was highly controlled by Sun may have created risk for Google in terms of Google's ability to move into the market quickly and to take development of the platform wherever Google wished. Similarly, a company in Google's position would not be amenable to open source licensing terms that would create risks for potential OEM customers who might have to relinquish control over changes that they made to the platform. Given that Google was attempting to get initial adoption of the platform, such friction would have been unattractive from a business sense.

236. I have reviewed some of the history of Sun's and Google's interactions regarding an "open source" GPL license, and public documents reflecting Google's position about that license at the time that it was attempting to introduce Android into the market. It is my opinion that Google did not want take such a license because important potential device manufacturer partners likely would not risk accepting code encumbered by this type of license. Again, this behavior is consistent with a party introducing a new platform that needed to be widely adopted by partners. Based on the fact that Google's behavior was consistent with a party attempting to create and quickly bring to market a new platform, it is my opinion that Google would not at the time have found Sun's "open source" license to be a commercially acceptable substitute for the unauthorized copying that Google ultimately carried out. There is historical evidence that this was, in fact, the case.

237. Additional documents show that Google was not willing to accept the "open source" license that Sun offered for Java, which is called the GPLv2 license with Classpath exception (here simply the "GPL" license). During negotiations in February 2006, Sun offered this GPL license to Google, but Google did not accept it. OAGOOGLE0001817230.[224]

[224] Sun and Oracle eventually offered the GPL licensed code as part of a project called "OpenJDK," in May 2007. *See* http://mail.openjdk.java.net/pipermail/announce/2007-May.txt

238. In a November 13, 2006 email, Google employee Cedric Beust[225] emailed members of the Android team and commented on "GPL2 + Classpath exception for the SE libraries" observing that "I would already be happy if we can just drop in their libraries, but still not sure it's compatible with their licensing choices."[226]

239. For example, in an August 11, 2007 email, Andy Rubin, Google's head of Android, explained to key Android engineers why Google could not use Sun's version of Java licensed under the GPL license: "...and as far as GPL-ing the VM, everything that is linked with the VM would get infected. The problem with GPL in embedded systems is that it's viral, and there is no way (for example) OEMs or Carriers to differentiate by adding proprietary works. We are building a platform where the entire purpose is to let people differentiate on top of it. Finally, Sun has a different license for its library for SE and ME. The SE library is LGPL, ME library is GPL. That means anything that links with the ME library gets infected. And the SE library is not optimized for embedded systems. Sun chose GPL for this exact reason so that companies would need to come back to them and take a direct license and pay royalties. Tricky, no?"[227]

240. In an August 11, 2007 email exchange between Andy Rubin (Android's founder) and Bob Lee (Google's former Core Library Lead for Android), Rubin comments that GPLv2 with the Classpath Exception licensed Java would not work due to concerns regarding Android OEMs and carriers. Rubin stated: "Here it is from their FAQ: Is the entire OpenJDK code base under the GPL? A: Yes, much of the OpenJDK code is licensed under the GPL version 2 with the Classpath exception. The Java Hotspot virtual machine source code is licensed under the GPL version 2 only. It is explained by Sun quite concisely here: http://forums.java.net/jivethread.jspa?messageID=175849#175849 *Bottom line: we do not want carriers or OEMs to be required to go to Sun if they are customers of our platform.*" (emphasis added)[228]

241. On January 4, 2007, Dan Bornstein, one of the primary Android engineers, expressed the view that the GPL license "makes it impossible to create 'extended subsets' of the libraries based on the GPLed version" of Java. GOOGLE-02-00079838 (Ghuloum Depo. Ex. 5018). Google's witness regarding the viability of GPL code admitted that as of 2007, Google was concerned about applying GPL to Android: "Q So it's correct that this January 4, 2007, e-mail from Dan Bornstein to others at Google is discussing the GPL license,

---

[225] An engineer specifically hired to assist with Google's Java efforts. *See* Google Continues to Grow Top Notch Java Team with Cedric Beust, D. Almaer, ServerSide, http://www.theserverside.com/news/thread.tss?thread_id=28264

[226] GOOGLE-01-00025477-78

[227] TX 230, GOOGLE-02-00020474.

[228] GOOGLE-01-00028498-99 (In general, there was and is significant uncertainty regarding the application of GPLv2 with Classpath Exception to Android); *see also* GOOGLE-02-00298859 (As Dan Bornstein put it in March 2007, given the uncertainty of the license obligations "why tempt fate?"); and, GOOGLE-03404608-GOOGLE-03404614 (Similarly, in May 2008, an attendee at Google I/O recounted Google's explanation for why it would not use GPLv2 with Classpath Exception: "The first and main reason they give for using Harmony instead of OpenJDK is the GNU license (GPL). Cell phone makers want to link proprietary value-add code directly into the system (into JVM-based apps. and/or service processes), and they do not want to worry about copyleft. Perhaps there is some education needed here about the class path exception. (I know I don't understand it; maybe they don't either. *And*, their license wonks appear to have a well-considered preference for Apache 2 over GPL+CPE.)")

correct?... THE WITNESS: It appears to be discussing GPL license. Q And near the bottom of the e-mail about two-thirds of the way down, Mr. Bornstein writes: "This makes it possible -- impossible to create 'extended subsets' of the libraries based on GPL'd version, which is something that Sun has historically feared." Do you see that? A Yes. Q So is it your understanding on behalf of Google that this e-mail is discussing making it impossible to create extended subsets of the Java libraries based on the GPL version of Java?... THE WITNESS: They appear to be discussing that. … Q And so don't you agree that it's correct that as of 2007, Mr. Bornstein was expressing concern about the GPL as applied to the Java libraries?... THE WITNESS: Yes, that does appear to be the case." (Ghuloum 30(b)(6), Anwar, 130:12-131:19, Dec. 9, 2015)

242. An April 2009 internal Google document was provided to Google developers and acted as a set of guidelines on the use of free or open source licensed code. The document noted that as Google made more products that were actually distributed (such as Android), the risks of open source licenses became greater. The document states that "Google engineers need to be careful to avoid using third-party software in Google products in ways that would cause Google to be required to reveal proprietary source code in order to comply with the license of such third-party software." The document warns against incorporating such software into Google products, and describes the licenses for the third party software as "restricted": "The 'restricted' licenses are the primary reason for the creation of this project. Licenses in this category require mandatory source distribution (including Google source code) if Google ships a product that includes third-party code protected by such a license. Also, any use of source code under licenses of this type in a Google product will "taint" Google source code with the undesirable license. Third-party software made available under one of these licenses must not be part of Google products that are delivered to outside customers. Such prohibited distribution methods include 'client ' (downloadable Google client software) and 'embedded' (such as software used inside the Google Search Appliance)."

- BCL
- Creative Commons licenses
- GNU Classpath's GPL + exception
- GNU GPL
- GNU LGPL (non C/C++ projects)
- NPL 1.0 and NPL 1.1
- OSL
- OPL
- Sleepycat License
- qmail Terms of Distribution
- Anything else not explicitly listed without prior legal sign-off.[229]

243. From the above list it can be seen that contemporaneously (2009) Google considered the "GNU Classpath's GPL + exception" as high risk. This is the same license that applied to Sun's OpenJDK code.

[229] GOOGLE-00303456-GOOGLE-00303472

244. In my opinion, Google would not take the GPL licensed version of Java as the basis for Android because it could be expected (and create risk) that a GPL licensed version of Java would have required Google and Android phone OEMs (such as Samsung, HTC and LG) to contribute back to the open source community (and offer under a GPL license) modifications and additions that they made to the API packages, as well as other derivative works that they created containing portions of the GPL licensed code (or at least there is evidence that there was sufficient concern with respect to these requirements).[230]

245. In this way, modifications or derivatives of the APIs in the core libraries created by phone OEMs would, themselves, be subject to the terms of the GPL license and could not be withheld by the phone OEMs as proprietary - product differentiation being part of OEM competition. There is considerable evidence that OEMs wanted to keep as closed source their customized drivers that exist in the hardware abstraction libraries in the user space of the Android OS stack. For example, in May 2008, Andy Rubin stated in an interview that he avoided GPL because of the risk that OEMs would have to contribute back modifications that they wanted to keep proprietary:

> "The thing that worries me about GPL is this: suppose Samsung wants to build a phone that's different in features and functionality than (one from) LG. If everything on the phone was GPL, any applications or user interface enhancements that Samsung did, they would have to contribute back … At the application layer, GPL doesn't work."[231]

246. At that time, for example, one industry observer noted: "A look at the Samsung mobile Web page reveals that there is a proprietary API that may be made available to developers."[232]

247. And, in fact, it was recently reported (since my initial report) in the *Wall Street Journal* that Samsung does suffer when it cannot differentiate its product:

> "Samsung, originally a maker of components for Apple, tried to move up the stack by creating its own cellphones. At first it succeeded, but lately it has foundered on its inability to differentiate its products by creating its own software and is now increasingly overrun by competitors taking advantage of the commodification of Android smartphones."[233]

---

[230] Ghuloum Deposition Tr., 32:1-33:25 (For example, if Google or an OEM made additions to the core library APIs, those would have to be licensed under GPL. Similarly, OEMs may make changes to the application frameworks or native libraries portions of the Java platform. Further, the Android platform stack has sets of APIs called the *application frameworks* and the *libraries*, in addition to the core libraries containing the 37 Java API packages.) *see also*, Ghuloum Deposition Tr., 41:8-43:16, 125:16-129:14 (For example, testifying that APIs in the applications frameworks may create subclasses of classes in the core libraries or implement interfaces in the core libraries) These can be seen in the representation of the Android platform at Figure 13. Phone manufacturers, such as Samsung or HTC, may want to make additions or changes to APIs in the application framework or libraries in Android, which extend, implement or otherwise modify or combine with the APIs in the core libraries.

[231] Google Carves an Android Path through Open-Source World, S. Shankland, CNET, http://www.cnet.com/news/google-carves-an-android-path-through-open-source-world/ (accessed Feb. 8, 2016)

[232] Federico C. Gonzalez, Entreprenews; Jumping the Curve: Location, location, location, *BusinessWorld*, September 4, 2008.

[233] Mims, C., "Why Big Companies Keep Getting Disrupted", *Wall Street Journal*, January 25, 2016.

248. Likewise, Patrick Brady, a Google Engineer, stated in a 2008 Google I/O Presentation: "Kernel drivers are GPL, which exposes IP. They [hardware manufacturers] want to keep theirs [drivers] closed source. We needed to provide a way to do that in the user space outside of the Linux kernel."[234] Similarly, Rich Miner, Android Co-Founder and Google Ventures Partner stated in 2007: "We don't open source any of the baseband code, and there is a very restricted protocol for communicating back and forth from the application processor over to the baseband side of the phone. We work pretty hard with our OEMs to make sure the baseband side is well-protected. They won't be exposing or documenting that part of the system."[235] Enabling OEMs to hold their changes to Android as proprietary is important in the market for Android, because without the ability to do so, it is very difficult for handset makers like Samsung and HTC to differentiate their products and compete. As discussed in my initial report, allowing this type of differentiation was critical in enabling Google to get OEM support when it first released Android in 2007. The need for these OEMs to differentiate based on their proprietary changes to Android continues today and is frequently documented in the industry.[236]

249. If Google had taken the GPL licensed version of Java, but Google or its phone manufacturer customers had tried to hold back any modifications that they made to Android, I have been informed that they would face the risk of a copyright lawsuit, since that activity could cause the license to terminate.[237] Thus, Google refused to accept the GPLv2 license that was offered by Sun. A Sun executive has testified that Sun was willing to contemplate doing a deal with Google under a GPL license because he understood it would serve as an incentive for Google to take a commercial, revenue generating commercial license instead.[238] Internal Sun documents from 2006 indicate that simultaneously offering a commercial license and a GPL license for Java would create a dual-licensing framework that would, among other things: "Reduce the likelihood of incompatible forks

---

[234] *See* Google I/O 2008 - Anatomy and Physiology of an Android, Google Developers, at https://www.youtube.com/watch?v=G-36noTCaiA, at 31:17 (accessed Feb. 8, 2016)

[235] Android: Building a Mobile Platform to Change the Industry, R. Miner, https://www.youtube.com/watch?v=WUrMI9ZGxQ8, at 1:04:42 (accessed Feb. 8, 2016)

[236] Samsung's Mobile OS Dilema, Monday Note, http://www.mondaynote.com/2015/06/22/samsungs-mobile-os-dilemma/; Battle of the Androids: Google Android vs. Samsung Android, *CS News*, http://www.osnews.com/story/26751/Battle_of_the_Androids_Google_Android_vs_Samsung_Android; HTC confirms some current devices will get Sense 5, or at least parts of it, Android Central, http://www.androidcentral.com/htc-confirms-some-current-devices-will-get-sense-5-or-least-parts-it; Everything you need to know about Sense 6 on the HTC One (M8), R. Nazarian, Talk Android, http://www.talkandroid.com/guides/htc-one-m8-guides/everything-you-need-to-know-about-sense-6-on-the-htc-one-m8/ (all accessed Feb 8, 2016)

[237] For discussion of lawsuits brought against software companies for failing to comply with the GPL license, *see* "Lawsuit threatens to break new ground on the GPL and software licensing issues," OpenSource.com, https://opensource.com/law/14/7/lawsuit-threatens-break-new-ground-gpl-and-software-licensing-issues; "VMware sued for alleged GPL license infractions," J. Kirk, *PC World*, http://www.pcworld.com/article/2893852/vmware-sued-for-alleged-gpl-license-infractions.html; and "The Story of BusyBox and The First GPL Lawsuit," J. Saddington, *Torque*, http://torquemag.io/busybox/ (all accessed Feb. 8, 2016)

[238] Gupta Deposition Tr., 189:3-189:20 ("Q. And so were you willing to do a deal with Google for an Open Source mobile platform without the licensing component? … THE WITNESS: No. Again, the licensing component is this license. We are partial owners of the platform, we are checking and checking out, it's our Open Source license model that we want, which means we have huge control of what people can do with a platform. Again, in Open Source model, if you read the whole thing, some people might assume that some people can ship the platform for free. The point is if we picked what we -- had what we picked after that anyway, which is GPL model into a Java platform, no OEM would ship. That means each one would be paying us per unit. So we would have preserved our existing model, we have got new investment in the platform and new business generation on the upside as well.)

gaining market traction," "Counters the potential for companies with large market power from creating incompatible or extended derivatives that establish lock-in," and "Makes Sun's commercial license + support more attractive to OEMs and licensees, retaining some revenue for Sun."[239] Sun also adopted GPLv2 with Classpath Exception in the OpenJDK project as an enforcement mechanism, to co-opt projects like Apache Harmony and keep those projects from taking off (or being adopted downstream).[240]

250. Google explicitly and publicly stated its reason for avoiding the GPL license when Android was released. For example, on Google's Android licensing page, Google points out that it chose to apply the more permissive Apache license to Android, instead of the LGPL license (a variant of the GPL license that I understand to be similar to GPLv2 with Classpath Exception) because the LGPL license would have caused business risk and uncertainty for OEM handset manufacturers and other business partners with respect to restrictions on their ability to withhold their software designs as proprietary. Google observes that handset manufacturers would be trying to avoid having the GPL terms apply to their own changes or designs, because those companies would not want to risk making their own code subject to the "viral" – as Google Android executives term it[241] – effect of the GPL license. In particular, the page states the following:

> "LGPL (in simplified terms) requires either: shipping of source to the application; a written offer for source; or linking the LGPL-ed library dynamically and allowing users to manually upgrade or replace the library. Since Android software is typically shipped in the form of a static system image, complying with these requirements ends up restricting OEMs' designs. (For instance, it's difficult for a user to replace a library on read-only flash storage.)"
>
> "LGPL requires allowance of customer modification and reverse engineering for debugging those modifications. Most device makers do not want to have to be bound by these terms. So to minimize the burden on these companies, we minimize usage of LGPL software in userspace."
>
> "Historically, LGPL libraries have been the source of a large number of compliance problems for downstream device makers and application developers. Educating engineers on these issues is difficult and slow-going, unfortunately. It's critical to Android's success that it be as easy as possible for device makers to comply with the licenses. Given the difficulties with complying with LGPL in the past, it is most prudent to simply not use LGPL libraries if we can avoid it."[242]

251. To understand the implications of these paragraphs, it is important to consider the concept of *userspace*. When Google refers to userspace it is speaking of all layers of code of a platform that are above the operating system "kernel."[243] In particular, in the case of Android, the kernel means the "Linux" kernel, and userspace

---

[239] OAGOOGLE0013819323-OAGOOGLE0013819333 at 326-327
[240] OAGOOGLE2000081380 (August 2005 Presentation, indicating need to deal with open source Apache Harmony Project and GNU Classpath/GCJ)
[241] TX 230; GOOGLE-02-00020474
[242] *See* Android Licenses," Android Open Source Project, http://source.android.com/source/licenses.html (accessed Feb. 8, 2015)
[243] https://en.wikipedia.org/wiki/User_space

means the code of all of the various APIs and application frameworks in the Android platform stack above the Linux kernel. (These can be seen in the representation of the Android platform at Figure 13. Google itself defines userspace as "non-kernel" software: "We are sometimes asked why Apache Software License 2.0 is the preferred license for Android. For userspace (that is, non-kernel) software, we do in fact prefer ASL2.0 (and similar licenses like BSD, MIT, etc.) over other licenses such as LGPL." https://source.android.com/source/licenses.html This suggests that Google wanted to protect all of the code in userspace from having to be GPL licensed. For example, a 2008 Google presentation discusses licensing issues regarding Android and acutely asserts: "we want to keep GPL out of user-space."[244]

252. Industry observers have explained that if the GPL license were applied to Android, it would cause difficulties because it would require OEM handset manufacturers to also apply the GPL license to their own modifications to Android code, thus preventing those companies from holding their own code back as proprietary. At the very least, applying GPL to Android would create significant uncertainty or risk for handset manufacturers in this regard. For example, a 2007 Ars Technica article by Ryan Paul, shortly after Android was released, observed that a major driver of Google's choice not to use the GPLv2 license was to protect the ability of handset makers and carriers to hold their modifications to Android as proprietary:[245]

> "The Apache license, chosen by Google "is widely used in the open-source software community and has been approved by the Open Source Initiative, is a permissive license that is conducive to commercial development and proprietary redistribution. Code that is distributed under the ASL and other permissive licenses can be integrated into closed-source proprietary products and redistributed under a broad variety of other terms. Unlike permissive open-source licenses, "copyleft" licenses (such as the GPL) generally impose restrictions on redistribution of code in order to ensure that modifications and derivatives are kept open and distributed under similar terms."[246]
>
> "Permissive licenses like the ASL and BSD license are preferred by many companies because such licenses make it possible to use open-source software code without having to turn proprietary enhancements back over to the open source software community. These licenses encourage commercial adoption of open-source software because they make it possible for companies to profit from investing in enhancements made to existing open-source software solutions. That potential for proprietary investment on top of an open stack is most likely what inspired Google to adopt the Apache Software License for its mobile platform. Availability of Android under the ASL will ensure that a broader number of companies will be able to adopt the platform and build on top of it without having to expose the inner workings of proprietary technologies that give them a competitive advantage."[247]

---

[244] GOOGLE-22-00280859-GOOGLE-22-00280977 at GOOGLE-22-00280894
[245] *See* Why Google Chose the Apache Software License over GPLv2 for Android, *Ars Technica*, http://arstechnica.com/uncategorized/2007/11/why-google-chose-the-apache-software-license-over-gplv2/(accessed Feb. 8, 2015)
[246] *Ibid.*
[247] *Ibid*

> "Although using a permissive license like ASL is the best way to build support for the Android platform, critics argue that Google has sacrificed an opportunity to encourage greater openness in the broader mobile software space. If Android was distributed under the GPLv2, companies building on top of the platform would have to share their enhancements, which could theoretically lead to widespread sharing of code and a more rapid acceleration of mobile software development."[248]
>
> "The counterargument is that distributing Android under a copyleft license could potentially limit the evolution of the mobile software ecosystem by discouraging commercial development on top of the platform. Proprietary mobile software development companies that integrate Android into their technologies would have to dramatically change their business models if they aren't given the ability to keep their enhancements proprietary."[249]
>
> "When it comes right down to it, the handset makers are the developers who are most significantly affected by the Android license, since they are the primary distributors of mobile phone platforms. The ASL will allow individual handset makers to develop proprietary customizations for the platform as needed to accommodate the unique technologies in their individual products."[250]

253. Google explicitly endorsed this explanation of why it would not accept the GPL license as applied to Android. On the Android website, Google stated: "Why are you releasing the code under the Apache License instead of GPLv2? One of the best explanations for the reasoning behind releasing code under Apache2 can be found in a ArsTechnica article by Ryan Paul," and linked to the article discussed above.[251] Internal discussion at Google shows that this article was approved by Andy Rubin, the head of Android. A November 11, 2007 IM exchange between Google employee David McLaughlin and Chris DiBona, the head of open source at Google, proposed the language endorsing the article and states that it was "suggested by andy yesterday" and "that was a great article." GOOGLE-14-00025664.

254. Another 2012 published article indicates that industry analysts did not believe that Google could release Android under the GPL license because "someone is going to have to write or port a lot of low-level code to access mobile specific capabilities related to smartphones. GPL licenses often restrict vendors from monetizing the code in certain ways without making it also open source, so that might be another factor."[252]

255. Based on the concerns expressed above, I believe that handset manufacturers in 2007-2010 would have resisted use and distribution of GPL licensed versions of Android, as it would limit the handset manufacturers' ability to withhold as proprietary their own modifications to Android and would therefore create a business

---

[248] *Ibid*
[249] *Ibid*
[250] *Ibid*
[251] *See* Open Source Licensing, Android Open Source Project, https://web.archive.org/web/20090124043917/http://code.google.com/android/kb/licensingandoss.html#apache2 (accessed Feb. 8, 2015)
[252] *See* Open source Java for Android? Don't Bet on it, P. Krill, *Java World*, http://www.javaworld.com/article/2078666/mobile-java/open-source-java-for-android--don-t-bet-on-it.html

risk for them. For example, a Google executive testified that Google would have to get OEMs to agree in advance that they were willing to subject any new APIs they created to GPL, but could not say with certainty that OEMs would agree to that. (Ghuloum 30(b)(6), Anwar, 35:14-36:3, 58:4-58:8, 58:17-59:10, 62:14-62:19 Dec. 9, 2015). The risks of OEM concerns would have been particularly acute at a time of initial introduction of the platform when its success was uncertain. As of December 2015 Google had not used GPL-licensed code in any prior version of Android. (Ghuloum 30(b)(6), Anwar, 27:24-28:5, 30:9-30:14, Dec. 9, 2015).

256. I have reviewed the report of Andrew Hall, which addresses open source licenses and the GPLv2 with Classpath Exception license in particular. Mr. Hall's report generally concludes that GPLv2 with Classpath Exception would have been a possibility. However, Mr. Hall entirely ignores and does not discuss at all any of the overwhelming and clear contemporaneous evidence that Google and its partners would not have accepted the GPLv2 with Classpath Exception license from 2005 and later in the early-mid 2000s, when Android was being developed and released. Mr. Hall's conclusions do not square with historic reality and fail to account for the actual risk environment and facts that influenced Google and its partners. For this reason, Mr. Hall's high level speculation that Google might have attempted to use GPLv2 with Classpath Exception licensed code for the 37 Java APIs is incorrect, as the historical documentation shows that Google would not and could not do so, given its understanding and perception of the risks.

257. Also, Mr. Hall does not analyze the actual technical and legal implications to Android, OEMs and carriers if 37 Java API packages under the GPLv2 with Classpath Exception were distributed in a version of Android. The failure to conduct a risk assessment, based on actual facts specific to Android or the circumstances of this case, means that Mr. Hall's conclusions are not based on an analysis of applicable, historic data.

258. I have reviewed the report of Dr. Schmidt which decompiles changes and additions that OEMs have made to their phone-specific versions of Android. (Schmidt Report, §X-E) I have also reviewed the Expert Report of Ms. Gwyn Murray, a lawyer who is an open source licensing expert. Ms. Murray has analyzed OEM-specific changes and additions to Android, as well as other parts of the Android stack created by Google that are not licensed under GPL. She has concluded that there are significant risks that the OEM changes and additions, and other portions of the Android platform not licensed under GPL would have to be open-sourced, if Android had incorporated GPLv2 with Classpath Exception-licensed versions of the 37 Java APIs into the core libraries. This is completely consistent with the risk assessment made by Google and OEMs for nearly a decade, as documented by Google's own materials and public commentary. In sum, it is my opinion that Google and OEMs would not, and could not, have accepted the GPLv2 with Classpath Exception licensed versions of the Java APIs. Mr. Hall's speculative opinions do not explicitly reflect these facts.

TABLE OF APPENDICES

APPENDIX A – Glossary of Terms

Acronym	Meaning	Description
CDC	Connected Device Configuration	▪ Configuration of Java ME for high end personal consumer and embedded devices ▪ Devices: Smart communicators, pagers, high-end PDAs, set top boxes[253]
CLDC	Connected Limited Device Configuration	▪ Sun's implementation / configuration of Java ME, designed for resource-constrained devices (non smartphones)[254] ▪ Devices: Mobile phones, pagers, mainstream PDAs[255]
GPLv2	GNU General Public License, version 2, with classpath exception	Version of the GNU General Public License, released in 1991[256]
J2ME	Java Platform Micro Edition	Same as JavaME[257]
J2SE	Java 2 Standard Edition	▪ Now: Java SE ▪ Java SE was called J2SE between versions 1.2-1.5[258]
Java ME	Java Platform Micro Edition	▪ Allows applications to run on mobile devices: micro-controllers, sensors, gateways, mobile phones, PDAs, TV set top boxes, printers, etc.…[259]
Java SE	Java Platform Standard Edition	▪ Allows users to develop and deploy Java applications on desktops and servers and embedded environments[260]

[253] *See, e.g.*, Java ME Technology – CDC, Oracle, http://www.oracle.com/technetwork/java/javame/tech/index-jsp-139293.html (last visited Feb 8, 2016)
[254] *See, e.g.*, Connected Limited Device Configuration (CLDC); JSR 139, Oracle, http://www.oracle.com/technetwork/java/cldc-141990.html (last visited Feb. 8, 2016)
[255] *See, e.g.*, Connected Limited Device Configuration (CLDC); JSR 139, Oracle, http://www.oracle.com/technetwork/java/cldc-141990.html (last visited Feb. 8, 2016)
[256] [256] *See* GNU General Public License, Version 2, GNU Operating System, http://www.gnu.org/licenses/old-licenses/gpl-2.0.en.html (last visited Feb 8, 2016)
[257] *See* What is J2ME or Java ME? Oracle, https://java.com/en/download/faq/whatis_j2me.xml (last visited Feb 8, 2016)
[258] *See* J2SE 1.4.2, Oracle, http://www.oracle.com/technetwork/java/javase/index-jsp-138567.html (last visited Feb 8, 2016)
[259] *See* Java Platform, Micro Edition (Java ME), Oracle, http://www.oracle.com/technetwork/java/embedded/javame/index.html (last visited Feb 8, 2016)
[260] *See* Java SE at a Glance, Oracle, http://www.oracle.com/technetwork/java/javase/overview/index.html (last visited Feb 8, 2016)

Acronym	Meaning	Description
JCP	Java Community Process	▪ Allows worldwide Java community to participate in the future of Java and APIs ▪ Established in 1998 ▪ The JCP is used in standardizing certain Java APIs in feature phones and defragment the Java ME market such that handset makers don't have to make handsets that run the software of specific software makers—they can download the standard Java APIs that'll do that
JDK	Java Development Kit	▪ Allows the writing of Java applications ▪ Includes: Java Runtime Environment, Java Compiler and the Java APIs[261]
JRE	Java Runtime Environment	▪ What you get when you download Java software ▪ Includes: JVM, Java platform core classes, supporting Java libraries[262]
JSR	Java Specification Requests	▪ Basically requests for specific changes to the Java language, libraries, and other components. JSRs are created as part of the Java Community Process, where interested parties can propose ideas for enhancements and the JCP executive committee will review and approve as appropriate ▪ They are APIs like Mobile 3D Graphics (JSR 184), Bluetooth (JSR 82), PDA Optional Packages (JSR 75), Wireless Messaging (JSR 205), Mobile Media (JSR 135), Advanced Multimedia Supplements (JSR 234) [263]
JVM	Java Virtual Machine	▪ Works with/on top of operating systems to run Java applications ▪ What allows Java programs to run on Windows and Mac[264]

[261] *See, e.g.*, Java™ Platform Standard Edition 7 Names and Versions, Oracle, http://www.oracle.com/technetwork/java/javase/jdk7-naming-418744.html

[262] *See* Java Programming Environment and the Java Runtime Environment (JRE), Oracle, https://docs.oracle.com/cd/E19455-01/806-3461/6jck06gqd/index.html

[263] *See, e.g.*, JSR Overview, Java Community Process, Java Community Protocol, https://jcp.org/en/jsr/overview

[264] *See* Java Virtual Machine, Business Dictionary, http://www.businessdictionary.com/definition/Java-virtual-machine-JVM.html (last visited Feb 8, 2016)

Acronym	Meaning	Description
MIDP	Mobile Information Device Profile	▪ MIDP is connected with CDC to create the "runtime environment" for mobile devices ▪ Provides core application functionality for mobile devices ▪ Asko Komsi (Nokia's Director of Industry Relations): In the early days of J2ME, MIDP was one of the APIs, nobody defined a framework. Carriers offered MIDP support on many handsets offered[265]
OpenJDK	Open Java Development Kit	An open-source implementation of Java SE. Companies and individuals license through the GPL model[266]
SOAP	Simple Object Access Protocol	A protocol specification for exchanging structured information in the implementation of web services in computer networks.
TCK	Technology Compatibility Kit	Test suite to show compatibility with Java[267]

[265] *See* Mobile Information Device Profile (MIDP); JSR 118, Oracle, http://www.oracle.com/technetwork/java/index-jsp-138820.html (last visited Feb 8, 2016)
[266] *See* OpenJDK, OpenJDK, http://openjdk.java.net/ (last visited Feb 8, 2016)
[267] *See* TCK Project Planning and Development Guide, Java Community Protocol, https://jcp.org/aboutJava/communityprocess/ec-public/TCK-docs/ppg.pdf, at pg. 2 (last visited Feb 8, 2016)

APPENDIX B – Materials Considered

Public Sources

- 17 U.S.C. § 106
- 17 U.S.C. § 107
- A Developer's-Eye View of Smartphone Platforms, P. Wayner, *InfoWorld*, http://www.infoworld.com/article/2675582/application-development/a-developer-s-eye-view-of-smartphone-platforms.html?page=2 (accessed Feb. 8, 2016)
- AdWords API Terms and Conditions, Google AdWords, http://web.archive.org/web/20050130011554/http://www.google.com/apis/adwords/terms.html (accessed Feb. 8, 2016)
- AdWords API Terms and Conditions, Google AdWords, http://web.archive.org/web/20060903142934/http://www.google.com/apis/adwords/terms.html (accessed Feb. 8, 2016)
- AdWords API Terms and Conditions, Google AdWords, https://web.archive.org/web/20140704175508/https://developers.google.com/adwords/api/docs/terms (nearly identical terms (accessed Feb. 8, 2016)
- AdWords API Terms and Conditions, Google AdWords, https://web.archive.org/web/20150406213931/https://developers.google.com/adwords/ api/docs/terms (accessed Feb. 8, 2016)
- AdWords API Terms and Conditions, Google AdWords, https://web.archive.org/web/20150922081038/https://support.google.com/adwordspolicy/ answer/6169371 (all accessed Feb. 8, 2016)
- Amazon Introduces API Gateway, Aims to Dominate Yet Another Area, B. Kepes, *Network World*, http://www.networkworld.com/article/2942602/cloud-computing/amazon-introduces-api-gateway-aims-to-dominate-yet-another-area.html (accessed Feb. 8 2016)
- An API for Healthcare Providers, *True Vault*, https://www.truevault.com/healthcare-api.html (accessed Feb. 8, 2016)
- Android Licenses," Android Open Source Project, http://source.android.com/source/licenses.html (accessed Feb. 8, 2015)
- Android: Building a Mobile Platform to Change the Industry, R. Miner, https://www.youtube.com/watch?v=WUrMI9ZGxQ8, at 1:04:42 (accessed Feb. 8, 2016)
- Apache Foundation to Vote Down Java 7, R.Paul, *Ars Technica*, http://arstechnica.com/information-technology/2010/11/apache-foundation-to-vote-down-java-7-protesting-oracle-abuses/ (accessed Feb. 8, 2016)
- Apache Harmony Project (archived), Apache Software Foundation, http://attic.apache.org/projects/harmony.html (accessed Feb. 8, 2016)
- API Craft Conference, http://www.apicraft.org/Speaker Deck, APIStrat, https://speakerdeck.com/apistrat (all accessed Feb. 8, 2016)
- API Days Conference, http://sf.apidays.io/ (accessed Feb. 8, 2016)
- API Strategy 201: Private APIs vs. Open APIs, API Academy, http://www.apiacademy.co/resources/api-strategy-lesson-201-private-apis-vs-open-apis/ (accessed Feb. 8, 2016)
- API World Conference, http://apiworld.co/conference/ (accessed Feb. 8, 2016)
- Apple Drops NDA for Released iPhone Software, B. Wilson, http://www.cnet.com/news/apple-drops-nda-for-released-iphone-software/ (accessed Feb. 8, 2016)
- Battle of the Androids: Google Android vs. Samsung Android, *CS News*, http://www.osnews.com/story/26751/Battle_of_the_Androids_Google_Android_vs_Samsung_Android (accessed Feb. 8, 2016)
- Berlind, D., "Long Live The Private API," *Programmable Web*, Feb. 6, 2015 (http://www.programmableweb.com/news/long-live-private-api/analysis/2015/02/06) (accessed Feb. 8, 2016)
- Bing Search API, Microsoft Azure Marketplace, http://datamarket.azure.com/dataset/bing/search#terms (accessed Feb 8, 2016)
- Bosch, J, "From software product lines to software ecosystems" in *Proceedings of the 13th International Software Product Line Conference*, August, 2009, Carnegie Mellon University
- Briggs, B., and C. Hodgetts, "Tech Trends 2015: The Fusion of Business and IT," *Supply Chain* 247 (2015), available at: http://dupress.com/periodical/trends/tech-trends-2015/?id=us:2el:3dc:dup:eng:cons:tt15 (accessed Feb. 8, 2016)

- Capture of Apollo Lunar Module Reliability Lessons Learned: Reliability Engineering, NASA, http://llis.nasa.gov/lesson/1835 (accessed Feb. 8, 2016)
- Connected Limited Device Configuration (CLDC); JSR 139, Oracle, http://www.oracle.com/technetwork/java/cldc-141990.html (last visited Feb. 8, 2016)
- Cusumano, Michael A. and David B. Yoffie, *Competing on Internet Time: Lessons from Netscape and its Battle with Microsoft*, New York: The Free Press, 1998
- Dortch, M., "Many Paths, One Mountain: PC-Mini Integration," *InfoWorld*, Nov. 27, 1989
- Evans, P. and R. Basole, "Revealing the API Ecosystem and Enterprise Strategy via Visual Analytics", *Communications of the ACM*, February 2016, v. 59, n. 2, p. 27 (accessed Feb. 8, 2016)
- Everything you need to know about Sense 6 on the HTC One (M8), R. Nazarian, *Talk Android*, http://www.talkandroid.com/guides/htc-one-m8-guides/everything-you-need-to-know-about-sense-6-on-the-htc-one-m8/ (accessed Feb 8, 2016)
- FAQ for the Bing Search API: Windows Azure Marketplace," Microsoft Corp., https://onedrive.live.com/view.aspx?resid=9C9479871FBFA822!110&app=Word&authkey=!AInKSlZ6KEzFE8k (accessed Feb. 8, 2016)
- Federico C. Gonzalez, Entreprenews; Jumping the Curve: Location, location, location, *Business World*, September 4, 2008.
- Five Actions for Maximizing Your API Value, C. Haddad, IT Briefcase, http://www.itbriefcase.net/five-actions-for-maximizing-your-api-value (accessed Feb. 8, 2016)
- Fleig, C.P., "Big Blue keeps a close eye on open architecture," *Network World*, Oct. 20, 1986, p. 1;also "NetBIOS - History and Terminology," Liquisearch, http://www.liquisearch.com/netbios/history_and_terminology (accessed Feb. 8, 2016)
- From AdWords to Firebase: The Road to API Nirvana, M. Waite, https://speakerdeck.com/apistrat/from-adwords-to-firebase-the-road-to-api-nirvana (accessed Feb. 8, 2016)
- Gaia Project Agrees To Google Cease and Desist," *SlashDot*, http://tech.slashdot.org/story/06/11/25/225256/gaia-project-agrees-to-google-cease-and-desist (accessed Feb. 8, 2016)
- GNU General Public License, Version 2, GNU Operating System, http://www.gnu.org/licenses/old-licenses/gpl-2.0.en.html (last visited Feb 8, 2016)
- Goncalves, V. and P. Ballon, "An exploratory analysis of Software as a Service and Platform as a Service models for mobile operators," in "Intelligence in Next Generation Networks," *ICIN* 2009, vol., no., pp.1-4, 26-29, Oct. 2009
- Google blocks Windows Phone YouTube app (again), for 'manufactured' reasons, P. Bright, *Ars Technica*, http://arstechnica.com/gadgets/2013/08/google-blocks-windows-phone-youtube-app-again-for-manufactured-reasons/;
- Google Carves an Android Path through Open-Source World, S. Shankland, *CNET*, http://www.cnet.com/news/google-carves-an-android-path-through-open-source-world/ (accessed Feb. 8, 2016)
- Google Continues to Grow Top Notch Java Team with Cedric Beust, D. Almaer, *ServerSide*, http://www.theserverside.com/news/thread.tss?thread_id=28264 (accessed Feb. 8, 2016)
- Google Demands Microsoft Pull YouTube App from Windows Phone, M. Honan, *Wired*, http://www.wired.com/2013/05/google-msft-youtube/ (accessed Feb. 8, 2016)
- Google I/O 2008 - Anatomy and Physiology of an Android, Google Developers, at https://www.youtube.com/watch?v=G-36noTCaiA, at 31:17 (accessed Feb. 8, 2016)
- Google is Forcing Routebuilder to Shut Down, A.Martin, *Medium*, https://medium.com/hacker-daily/google-maps-is-forcing-routebuilder-to-shutdown-615ce42f413a#.q467avy9j (accessed Feb. 8, 2016)
- Google Maps APIs Pricing and Plans, Google Developers, https://developers.google.com/maps/pricing-and-plans/?hl=en (accessed Feb 8, 2016)
- Configuration and Reporting API Limits and Quotas, Google Developers, https://developers.google.com/analytics/devguides/reporting/mcf/v3/limits-quotas (accessed Feb 8, 2016)
- Google Maps/Google Earth APIs Terms of Service, Google Developers, https://developers.google.com/maps/terms?hl=en
- Google threatens to sue Youtube to MP3 conversion web site, L. Bell, *The Inquirer*, http://www.theinquirer.net/inquirer/news/2185556/google-threatens-sue-youtube-mp3-conversion-web-site (accessed Feb. 8, 2016)
- Harmony! M. Wielaard, GMane.org, http://article.gmane.org/gmane.comp.java.classpath.devel/5521 (accessed Feb. 8, 2016)
- High-Productivity Software for Complex Networks, NetworkX, http://networkx.github.io/ (accessed Feb. 8, 2016)

- History of UNIX, R. Hauben, http://minnie.tuhs.org/TUHS/Mirror/Hauben/unix.html, http://minnie.tuhs.org/TUHS/Mirror/Hauben/unix-Part_II.html
- HTC confirms some current devices will get Sense 5, or at least parts of it, *Android Central*, http://www.androidcentral.com/htc-confirms-some-current-devices-will-get-sense-5-or-least-parts-it (accessed Feb. 8, 2016)
- I Love APIs Conference, http://iloveapis.com/ (accessed Feb. 8, 2016)
- Iansiti, Marco, and Roy Levien, The Keystone Advantage: What the New Dynamics of Business Ecosystems Mean for Strategy, Innovation, and Sustainability, Boston: Harvard Business School Press, 2004
- Identifying Critical Attack Assets in Dependency Attack Graphs, R.Sawilla and X. Ou, http://people.cis.ksu.edu/~xou/publications/drdc08.pdf (accessed Feb. 8, 2016)
- If You Love Something, Set It Free, T. Sweeney, Unreal Engine, https://www.unrealengine.com/blog/ue4-is-free
- If You Love Something, Set It Free, T. Sweeney, Unreal Engine, https://www.unrealengine.com/blog/ue4-is-free (accessed Feb. 8 2016)
- Introduction to Social Network Methods, R. A. Hanneman, http://faculty.ucr.edu/~hanneman/nettext/C10_Centrality.html (accessed Feb. 8, 2016
- J2SE 1.4.2, Oracle, http://www.oracle.com/technetwork/java/javase/index-jsp-138567.html (last visited Feb 8, 2016)
- Java 2 Platform Standard Edition Development Kit 5.0 Specification, Oracle, http://docs.oracle.com/javase/1.5.0/docs/relnotes/license.html (accessed Feb. 8, 2016)
- Java Community Protocol, https://jcp.org/en/participation/members/ (accessed Feb. 8, 2016)
- Java ME Technology – CDC, Oracle, http://www.oracle.com/technetwork/java/javame/tech/index-jsp-139293.html (last visited Feb 8, 2016)
- Java Platform, Micro Edition (Java ME), Oracle, http://www.oracle.com/technetwork/java/embedded/javame/index.html (last visited Feb 8, 2016)
- Java Programming Environment and the Java Runtime Environment (JRE), Oracle, https://docs.oracle.com/cd/E19455-01/806-3461/6jck06gqd/index.html (accessed Feb. 8, 2016)
- Java SE 5.0 Downloads," Oracle, http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-javase5-419410.html (accessed Feb. 8, 2016)
- Java SE at a Glance, Oracle, http://www.oracle.com/technetwork/java/javase/overview/index.html (last visited Feb 8, 2016)
- Java Virtual Machine, Business Dictionary, http://www.businessdictionary.com/definition/Java-virtual-machine-JVM.html (last visited Feb 8, 2016)
- JavaTM Platform Standard Edition 7 Names and Versions, Oracle, http://www.oracle.com/technetwork/java/javase/jdk7-naming-418744.html (accessed Feb. 8, 2016)
- JSR Overview, Java Community Process, Java Community Protocol, https://jcp.org/en/jsr/overview (accessed Feb. 8, 2016)
- Landes, W. and R. Posner, "An Economic Analysis of Copyright Law", 18 J. Leg. Stud. 325, 325-33, 344-53 (1989) *available at* http://cyber.law.harvard.edu/IPCoop/89land1.html (accessed Feb. 8, 2016)
- Lawsuit threatens to break new ground on the GPL and software licensing issues, OpenSource.com, https://opensource.com/law/14/7/lawsuit-threatens-break-new-ground-gpl-and-software-licensing-issues;
- Leveson, N.G., "High-pressure steam engines and computer software," in Computer , vol.27, no.10, pp.65-73, Oct. 1994, available at http://sunnyday.mit.edu/papers/steam.pdf
- Louvel, J., "Community Debates API Specification Alternatives," *InfoQ*, Feb. 18, 2015. Available at: http://www.infoq.com/news/2015/02/api-alternatives (accessed Feb 8., 2016);also "
- Medrano, R., "Welcome to the API Economy," *Forbes*, Aug 29, 2012, available at: http://www.forbes.com/sites/ciocentral/2012/08/29/welcome-to-the-api-economy/#7fbea7e16d396df08abb6d39 (accessed Feb. 8, 2016)
- Method for Node Ranking in a Linked Database, US Patent No. 6285999 B1, http://www.google.com/patents/US6285999 (accessed Feb. 8, 2016)
- Mims, C., "Why Big Companies Keep Getting Disrupted", *Wall Street Journal*, January 25, 2016
- Mobile Information Device Profile (MIDP); JSR 118, Oracle, http://www.oracle.com/technetwork/java/index-jsp-138820.html (last visited Feb 8, 2016)
- Mowery, D.C., (Ed.), *The International Computer Software Industry: A Comparative Study of Industry Evolution and Structure*, Oxford University Press, New York, 1996

- Network-Based Analysis of Software Change Propagation, R. Wang, R. Huang and B. Qu, http://www.hindawi.com/journals/tswj/2014/237243/ (accessed Feb. 8, 2016)
- O'Reilly, T., "What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software," *Communications & Strategies*, No. 1, p. 17, Q1, 2007
- Open APIs, Mashups and User Innovation, M. Weiss, http://timreview.ca/article/168 (accessed Feb. 8, 2016)
- Open or Closed API: Six Guidelines to Help CIOs Make the Call," L. Lawson, *Programmable Web*, http://www.programmableweb.com/news/open-or-closed-api-six-guidelines-to-help-cios-make-call/2012/08/29 (accessed Feb. 8, 2016)
- Open Source Java for Android? Don't Bet on It, P. Krill, Java World, http://www.javaworld.com/article/2078666/mobile-java/open-source-java-for-android--don-t-bet-on-it.html (accessed Feb. 8, 2016)
- Open Source Licensing, Android Open Source Project, https://web.archive.org/web/20090124043917/http://code.google.com/android/kb/licensingandoss.html#apache2 (accessed Feb. 8, 2015)
- OpenJDK, http://mail.openjdk.java.net/pipermail/announce/2007-May.txt (accessed Feb. 8, 2016)
- OpenJDK, OpenJDK, http://openjdk.java.net/ (last visited Feb 8, 2016)
- Oracle Java Archive, Oracle, http://www.oracle.com/technetwork/java/javase/archive-139210.html (accessed Feb 8, 2016)
- Oracle v. Google ruling shows why cloud players may have steered clear of Amazon APIs, B. Darrow, https://gigaom.com/2014/05/09/oracle-v-google-ruling-shows-why-cloud-players-may-have-steered-clear-of-amazon-apis/ (accessed Feb. 8, 2016)
- Page T+B198:B210rial Tr.
- Page, L., "The PageRank Citation Ranking: Bringing Order to the Web," *available at* http://ilpubs.stanford.edu:8090/422/1/1999-66.pdf (accessed Feb. 8, 2016)
- PageRank Centrality, http://www.sci.unich.it/~francesc/teaching/network/pagerank (accessed Feb. 8, 2016)
- PPC Platform Competition and Google's "May Not Copy" Restriction, B. Edelman, http://www.benedelman.org/news/062708-1.html#dataportability (summarizing rationales for the license restrictions offered by Doug Raymond, product manager for AdWords API) (accessed Feb. 8, 2016)
- Prediction API Pricing, Google Cloud, https://cloud.google.com/prediction/#pricing (all accessed Feb. 8, 2016)
- Proprietary APIs: A New Tool in the Age of the Platform," P. Simon, http://www.huffingtonpost.com/phil-simon/proprietary-apis-a-new-to_b_6061722.html (accessed Feb. 8 2016)
- Puppin, D. and F. Silvestri, "The Social Network of Java Classes," in *Proceedings of the 2006 ACM Symposium on Applied Computing*, pp. 1409-1413, ACM, 2006, available at http://pomino.isti.cnr.it/~silvestr/wp-content/uploads/2011/02/sac2006.pdf (accessed Feb. 8, 2016)
- Ranade, P., D. Scannell and B. Stafford, "Ready for APIs? Three steps to unlock the data economy's most promising channel," *Forbes*, Jan. 7, 2014 available at: http://www.forbes.com/sites/mckinsey/2014/01/07/ready-for-apis-three-steps-to-unlock-the-data-economys-most-promising-channel/#51a1d71e89e5 (accessed Feb. 8, 2016)
- Riggins, J., How To Design Great APIs With API-First Design and RAML, *Programmable Web*, Jul. 10, 2015. Available at: http://www.programmableweb.com/news/how-to-design-great-apis-api-first-design-and-raml/how-to/2015/07/10 (accessed Feb. 8, 2016)
- Salus, P., A Quarter Century of UNIX, Addison-Wesley, 1994
- Samsung's Mobile OS Dilema, Monday Note, http://www.mondaynote.com/2015/06/22/samsungs-mobile-os-dilemma/ (accessed Feb. 8, 2016)
- SciTools Understand, https://scitools.com/features/#feature-category-dependency-analysis (accessed Feb. 8, 2016)
- Speaker Deck, APIStrat, https://speakerdeck.com/apistrat (accessed Feb. 8, 2016)
- Steinmueller, W. E., ""The US Software Industry: An Analysis and Interpretive History," in *The International Computer Software Industry*, D.C. Mowery (ed.), Oxford University Press, 1995
- Students and teachers: You may be eligible to get Office for free!" Miscrosoft Office Blogs, https://blogs.office.com/2014/09/22/students-teachers-may-eligible-get-office-free/ (accessed Feb. 8 2016)
- Students, teachers and academic institutions worldwide are eligible for free* access to Autodesk software. Yes, free. We genuinely believe in education. Autodesk, http://www.autodesk.com/education/home (accessed Feb. 8 2016)
- Sun Liberates JDK, Delivers on Open-Source Java Promise, R. Paul, *Ars Technica*, http://arstechnica.com/uncategorized/2007/05/sun-liberates-jdk-delivers-on-open-source-java-promise/(accessed Feb. 8, 2016)

- Sun, Microsoft Settle Java Lawsuit, J. Niccolai, Java World, http://www.javaworld.com/article/2074908/sun--microsoft-settle-java-lawsuit.html (accessed Feb. 8, 2016)
- TCK Project Planning and Development Guide, Java Community Protocol, https://jcp.org/aboutJava/communityprocess/ec-public/TCK-docs/ppg.pdf, at pg. 2 (last visited Feb 8, 2016)
- The API economy journey map: How are you doing? IBM Systems, https://www.ibm.com/blogs/systems/the-api-economy-journey-map-how-are-you-doing/ (accessed Feb. 8, 2016)
- The Limits of Google's Openness, Microsoft Corp., http://blogs.microsoft.com/on-the-issues/2013/08/15/the-limits-of-googles-openness/ (all accessed Feb 8, 2016)
- The Story of BusyBox and The First GPL Lawsuit, J. Saddington, Torque, http://torquemag.io/busybox/ (all accessed Feb. 8, 2016)
- Threatens To Sue Huge YouTube MP3 Conversion Site, *Torrent Freak*, https://torrentfreak.com/google-threatens-to-sue-huge-youtube-mp3-conversion-site-120619/ (accessed Feb. 8, 2016);
- Transforming Global Information and Communication Markets, Information Revolution and Global Politics Series, MIT Press, 2012
- Translate API Pricing, Google Cloud, https://cloud.google.com/translate/v2/pricing;
- U.S. Const., Art. I, § 8
- UE4 Libraries You Should Know About, B. Bramer, *Unreal Engine*, https://www.unrealengine.com/blog/ue4-libraries-you-should-know-about (accessed Feb. 8, 2016)
- Unreal® Engine End User License Agreement, *Unreal Engine*, https://www.unrealengine.com/eula (accessed Feb. 8, 2016)
- User Space Definition, https://en.wikipedia.org/wiki/User_space (accessed Feb 8., 2016)
- VMware sued for alleged GPL license infractions, J. Kirk, PC World, http://www.pcworld.com/article/2893852/vmware-sued-for-alleged-gpl-license-infractions.html
- Welch , K. P., "Interprogram communications using Windows' dynamic data exchange," *Microsoft Systems Journal*, 2.5 (Nov. 1987)
- West, J., J. Dedrick, "Innovation and control in standards architectures: the rise and fall of Japan's PC-98," *Inform. Syst. Res.* 11 (2), 2000
- What is J2ME or Java ME? Oracle, https://java.com/en/download/faq/whatis_j2me.xml (last visited Feb 8, 2016)
- What's the best way to write an API spec? *Y-Combinator*, https://news.ycombinator.com/item?id=8912897 for more detailed interactions on the specific debates concerning API design tools and documentation procedures (accessed Feb 8. 2016)
- Why Google Chose the Apache Software License over GPLv2 for Android, *Ars Technica*, http://arstechnica.com/uncategorized/2007/11/why-google-chose-the-apache-software-license-over-gplv2/(accessed Feb. 8, 2016)
- Willmott, S. and G. Balas, "Winning in the API Economy," *available at* http://www.3scale.net/wp-content/uploads/2013/10/Winning-in-the-API-Economy-eBook-3scale.pdf (accessed Feb. 8 2016)
- Xcode and Apple SDKs Agreement, Apple Inc., https://www.apple.com/legal/sla/docs/xcode.pdf (accessed Feb. 8, 2016)

Produced Documents

- ASF-GOOGLE-00001393
- ASF-GOOGLE-00002148
- ASF-GOOGLE-00002863
- ASF-GOOGLE-00003030
- GOOG-00138478
- GOOG-00259405
- GOOG-00259405
- GOOG-00259405
- GOOG-00475452
- GOOG-00527347-GOOG-00527349;
- GOOG-00527350-GOOG-00527356
- GOOGLE-00303456-GOOGLE-00303472

- GOOGLE-00395207–GOOGLE-00395248;
- GOOGLE-01-00025477-78
- GOOGLE-01-00028498-99
- GOOGLE-02-00020474.
- GOOGLE-02-00298859
- GOOGLE-02-00384174
- GOOGLE-03374594–GOOGLE-003374599
- GOOGLE-03404608-GOOGLE-03404614
- GOOGLE-13-00171678
- GOOGLE-14-00002326
- GOOGLE-14-00003518
- GOOGLE-14-00039843
- GOOGLE-22-00280859-GOOGLE-22-00280977
- GOOGLE-22-00489760
- GOOGLE-22-00489771
- GOOGLE-24-00010460
- GOOGLE-24-00015413
- GOOGLE-24-00019558
- GOOGLE-40-00003728; GOOGLE-02-00298870
- GOOGLE-53-00319805
- OAGOOGLE0000494636
- OAGOOGLE0000609523
- OAGOOGLE0007356223
- OAGOOGLE0011736496-98
- OAGOOGLE0013819323-OAGOOGLE0013819333
- OAGOOGLE0020194367-68
- OAGOOGLE0025475455
- OAGOOGLE0029214822
- OAGOOGLE0100031837
- OAGOOGLE0100036648-OAGOOGLE0100036679
- OAGOOGLE0102318356
- OAGOOGLE2000081380
- OAGOOGLE2001355659
- OAGOOGLE2009765998
- TX 1
- TX 7
- TX 10
- TX 15
- TX 18
- TX 23
- TX 158
- TX 181
- TX 215
- TX 230
- TX 405

- TX 623
- TX 749
- TX 828
- TX 917
- TX 1045
- TX 1047
- TX 4002

Court Documents

- Bloch Trial Tr.
- Cattell Opening Report
- Lee Trial Tr.
- Morrill Trial Tr.
- Schmidt Opening Report
- Schwartz Trial Tr.
- United States Court of Appeals for the Federal Circuit, Oracle America Inc. v. Google Inc. 2013-1021, 1022
- Opening Expert Report of Dr. Owen Astrachan on Technical Issues Relating to Fair Use, United States District Court, Northern District of California, San Francisco Division, *Oracle America Inc. v. Google Inc.* 3:10-cv-03561-WHA, filed Jan. 8, 2016 ("Astrachan Opening Report," "Astrachan Report")
- Opening Report of Roderic G. Cattell, Ph.D., United States District Court, Northern District of California, San Francisco Division, *Oracle America Inc. v. Google Inc.* 3:10-cv-03561-WHA, filed Jan. 8, 2016 ("Cattell Opening Report," "Cattell Report")
- Expert Report of Professor Douglas C. Schmidt, Ph.D., United States District Court, Northern District of California, San Francisco Division, Oracle America Inc. v. Google Inc. 3:10-cv-03561-WHA, filed Jan. 8, 2016 ("Schmidt Opening Report," "Schmidt Report")
- Expert Report of Professor Douglas C. Schmidt, Ph.D Regarding Fair Use and Rebuttal to Google's Opening Expert Reports, United States District Court, Northern District of California, San Francisco Division, Oracle America Inc. v. Google Inc. 3:10-cv-03561-WHA, filed Jan. 8, 2016 ("Schmidt Rebuttal Report," "Schmidt Rebuttal")
- Expert Rebuttal Report of Gwyn Firth Murray United States District Court, Northern District of California, San Francisco Division, Oracle America Inc. v. Google Inc. 3:10-cv-03561-WHA, filed Jan. 8, 2016 ("Murray Rebuttal Report," "Murray Rebuttal," "Murray Report")

Depositions

- Brady Deposition Tr.
- Camargo Deposition Tr.
- Chu Deposition Tr.
- Ghuloum Deposition Tr.
- Gupta Deposition Tr.
- Holzle Deposition Tr.
- Lin Deposition Tr.
- Lockheimer Deposition Tr.
- Morrill Deposition Tr.
- Saab Deposition Tr.
- Smith Deposition Tr.

APPENDIX C – JDK PageRank Full Dataset[268]

PageRank scores for the copied classes in the JDK 5.0 source code[269], ordered from the highest score to the lowest score.

Class Name	PageRank Score
java.lang.String	0.140357
java.lang.Object	0.062873
java.lang.Exception	0.032709
java.lang.RuntimeException	0.026561
java.lang.Throwable	0.026141
java.lang.IllegalArgumentException	0.023693
java.lang.Class	0.020391
java.lang.Integer	0.016238
java.io.IOException	0.013669
java.io.Serializable	0.013490
java.lang.System	0.012875
java.lang.NullPointerException	0.012407
java.lang.Double	0.012254
java.lang.Float	0.011049
java.io.Bits	0.010668
java.lang.Math	0.010595
java.lang.Comparable	0.009454
java.lang.StringBuffer	0.009435
java.lang.IndexOutOfBoundsException	0.009179
java.lang.Character	0.009062
java.lang.CloneNotSupportedException	0.008464
java.lang.InterruptedException	0.007619
java.lang.CharSequence	0.006790
java.lang.Long	0.006748
java.lang.NumberFormatException	0.006739
java.lang.Number	0.006022
java.lang.Deprecated	0.005929
java.util.Locale	0.005900
java.io.ObjectStreamField	0.005617
java.lang.AbstractStringBuilder	0.005566
java.util.ArrayList	0.005178
java.lang.StringIndexOutOfBoundsException	0.005019

[268] For the 61 class sensitivity analysis, the PageRank scores of the 61 classes are removed from this dataset for further analysis; for the java.lang sensitivity analysis, the PageRanks scores of all the classes in java.lang package are removed from this dataset for further analysis.

[269] For JDK Java APIs, only publicly facing APIs are included since these are the only ones with data available for the PageRank analysis. This excludes the following 6 APIs from use in this analysis: javax.security.cert, javax.crypto, javax.crypto.interfaces, javax.crypto.spec, javax.net, and javax.net.ssl.

Class Name	PageRank Score
java.lang.StringBuilder	0.005006
java.io.ObjectStreamClass	0.004667
java.io.UnsupportedEncodingException	0.004418
java.util.regex.Matcher	0.004360
java.util.regex.Pattern	0.004360
java.lang.ConditionalSpecialCasing	0.004221
java.util.Formatter	0.004178
java.io.ObjectOutputStream	0.004089
java.util.Collection	0.004052
java.util.Iterator	0.004033
java.util.Comparator	0.004001
java.lang.StringCoding	0.003992
java.util.regex.PatternSyntaxException	0.003972
java.security.AccessController	0.003322
java.lang.ClassNotFoundException	0.003056
java.lang.IllegalStateException	0.002965
java.io.PrintStream	0.002944
java.io.ObjectInputStream	0.002557
java.util.Set	0.002262
java.util.Map	0.002178
java.util.Random	0.002091
java.lang.StackTraceElement	0.002066
java.util.Hashtable	0.001968
java.lang.Cloneable	0.001955
java.io.InputStream	0.001949
java.io.PrintWriter	0.001938
java.security.PrivilegedAction	0.001929
java.lang.StrictMath	0.001883
java.util.Enumeration	0.001785
java.lang.SecurityManager	0.001768
java.lang.Thread	0.001706
java.util.HashMap	0.001652
java.util.List	0.001603
java.lang.Error	0.001518
java.lang.InternalError	0.001480
java.util.ListResourceBundle	0.001451
java.lang.SecurityException	0.001336
java.io.OutputStream	0.001251
java.util.StringTokenizer	0.001181
java.security.AccessControlContext	0.001134
java.util.EventListener	0.001123
java.lang.Boolean	0.001113

Class Name	PageRank Score
java.io.File	0.001018
java.util.Vector	0.001014
java.util.AbstractCollection	0.001003
java.lang.reflect.Type	0.000996
java.util.HashSet	0.000986
java.lang.reflect.Array	0.000940
java.lang.reflect.TypeVariable	0.000933
java.lang.Iterable	0.000932
java.lang.ClassLoader	0.000921
java.security.Permission	0.000906
java.security.ProtectionDomain	0.000900
java.io.Closeable	0.000898
java.lang.reflect.Field	0.000873
java.util.ResourceBundle	0.000872
java.io.ObjectStreamException	0.000858
java.lang.Enum	0.000853
java.lang.ClassCastException	0.000816
java.lang.IllegalAccessException	0.000810
java.io.FileDescriptor	0.000784
java.lang.UnsupportedOperationException	0.000760
java.io.FileOutputStream	0.000750
java.util.EventObject	0.000746
java.lang.VirtualMachineError	0.000724
java.lang.Byte	0.000721
java.lang.Short	0.000718
java.util.Properties	0.000718
java.lang.reflect.Method	0.000697
java.lang.reflect.AccessibleObject	0.000693
java.util.PropertyPermission	0.000686
java.lang.NoSuchMethodException	0.000681
java.lang.Appendable	0.000669
java.net.URL	0.000665
java.lang.InstantiationException	0.000657
java.lang.annotation.Annotation	0.000651
java.util.MissingResourceException	0.000638
java.nio.Buffer	0.000626
java.util.NoSuchElementException	0.000613
java.lang.reflect.InvocationTargetException	0.000604
java.lang.reflect.GenericDeclaration	0.000603
java.lang.ThreadGroup	0.000589
java.lang.reflect.Modifier	0.000585
java.nio.ByteBuffer	0.000579

Class Name	PageRank Score
java.lang.reflect.Constructor	0.000569
java.lang.reflect.Member	0.000566
java.lang.ref.Reference	0.000565
java.io.FileInputStream	0.000554
java.lang.ref.SoftReference	0.000548
java.util.AbstractList	0.000537
java.io.Writer	0.000535
java.lang.ArrayIndexOutOfBoundsException	0.000530
java.lang.reflect.AnnotatedElement	0.000521
java.lang.RuntimePermission	0.000509
java.io.FileNotFoundException	0.000508
java.lang.CharacterData00	0.000498
java.lang.CharacterData01	0.000498
java.lang.CharacterData02	0.000498
java.lang.CharacterData0E	0.000498
java.lang.CharacterDataLatin1	0.000498
java.io.DataOutputStream	0.000497
java.lang.AbstractMethodError	0.000496
java.lang.Terminator	0.000476
java.io.BufferedInputStream	0.000460
java.security.AccessControlException	0.000454
java.util.Arrays	0.000449
java.io.DataInputStream	0.000446
java.lang.ref.ReferenceQueue	0.000440
java.security.PrivilegedExceptionAction	0.000434
java.io.BufferedOutputStream	0.000430
java.lang.NoSuchFieldException	0.000429
java.nio.channels.Channel	0.000428
java.text.BreakIterator	0.000427
java.util.Collections	0.000427
java.security.PrivilegedActionException	0.000423
java.nio.channels.spi.SelectorProvider	0.000416
java.security.AllPermission	0.000408
java.security.Permissions	0.000407
java.util.LinkedList	0.000407
java.text.MessageFormat	0.000399
java.util.LinkedHashSet	0.000396
java.io.BufferedWriter	0.000396
java.lang.Package	0.000395
java.util.ListIterator	0.000392
java.nio.BufferUnderflowException	0.000392
java.nio.BufferOverflowException	0.000392

Class Name	PageRank Score
java.lang.CharacterDataPrivateUse	0.000388
java.lang.CharacterDataUndefined	0.000388
java.lang.ProcessEnvironment	0.000380
java.lang.Runtime	0.000379
java.text.Format	0.000373
java.sql.SQLException	0.000361
java.io.OutputStreamWriter	0.000359
java.io.FilterOutputStream	0.000354
java.lang.reflect.GenericArrayType	0.000346
java.util.regex.MatchResult	0.000345
java.lang.LinkageError	0.000334
java.util.RandomAccess	0.000330
java.nio.ByteOrder	0.000329
java.lang.IncompatibleClassChangeError	0.000328
java.nio.CharBuffer	0.000327
java.security.BasicPermission	0.000323
java.nio.charset.Charset	0.000315
java.security.GeneralSecurityException	0.000314
java.util.IllegalFormatException	0.000313
java.nio.charset.CharacterCodingException	0.000313
java.lang.ThreadLocal	0.000312
java.io.StreamCorruptedException	0.000302
java.io.Flushable	0.000297
java.io.InterruptedIOException	0.000281
java.beans.PropertyChangeEvent	0.000276
java.net.InetAddress	0.000269
java.security.PermissionCollection	0.000266
java.util.regex.ASCII	0.000261
java.io.Reader	0.000261
java.text.CharacterIterator	0.000260
java.lang.Runnable	0.000257
java.util.Date	0.000255
java.io.SyncFailedException	0.000252
java.util.AbstractSet	0.000246
java.io.UTFDataFormatException	0.000243
java.io.DataOutput	0.000243
java.beans.PropertyChangeListener	0.000237
java.nio.Bits	0.000236
java.io.Externalizable	0.000234
java.nio.charset.CharsetEncoder	0.000233
java.util.ResourceBundleEnumeration	0.000219
java.io.ObjectStreamConstants	0.000218

Class Name	PageRank Score
java.util.ConcurrentModificationException	0.000217
java.nio.charset.CharsetDecoder	0.000215
java.io.SerializablePermission	0.000214
java.nio.ReadOnlyBufferException	0.000209
java.nio.charset.CoderResult	0.000207
java.io.FilterInputStream	0.000204
java.nio.charset.CodingErrorAction	0.000202
java.io.InvalidObjectException	0.000198
java.text.AttributedCharacterIterator	0.000183
java.nio.charset.MalformedInputException	0.000182
java.lang.NegativeArraySizeException	0.000182
java.nio.charset.UnsupportedCharsetException	0.000180
java.nio.charset.IllegalCharsetNameException	0.000180
java.lang.ref.WeakReference	0.000178
java.net.MalformedURLException	0.000177
java.io.CharConversionException	0.000176
java.security.Principal	0.000166
java.io.ObjectOutput	0.000162
java.util.Stack	0.000162
java.lang.reflect.Proxy	0.000161
java.io.ByteArrayOutputStream	0.000158
java.io.NotActiveException	0.000156
java.lang.Shutdown	0.000153
java.io.NotSerializableException	0.000152
java.nio.channels.spi.AbstractInterruptibleChannel	0.000150
java.io.DataInput	0.000147
java.security.Policy	0.000145
java.lang.ClassFormatError	0.000144
java.nio.channels.FileChannel	0.000137
java.security.Key	0.000133
java.io.InvalidClassException	0.000127
java.lang.ThreadDeath	0.000123
java.util.Calendar	0.000123
java.lang.reflect.ReflectAccess	0.000122
java.security.NoSuchAlgorithmException	0.000122
java.io.EOFException	0.000120
java.net.SocketException	0.000119
java.util.AbstractMap	0.000118
java.io.ByteArrayInputStream	0.000116
java.net.URI	0.000115
java.net.Socket	0.000114
java.util.TimeZone	0.000113

Class Name	PageRank Score
java.security.cert.Certificate	0.000112
java.lang.IllegalThreadStateException	0.000112
java.lang.AssertionError	0.000112
java.text.DecimalFormatSymbols	0.000110
java.text.NumberFormat	0.000109
java.io.ObjectInput	0.000107
java.util.Dictionary	0.000106
java.text.DecimalFormat	0.000105
java.sql.DriverManager	0.000104
java.net.UnknownHostException	0.000103
java.net.ServerSocket	0.000102
java.security.MessageDigest	0.000101
java.nio.InvalidMarkException	0.000100
java.nio.channels.SelectionKey	0.000098
java.sql.SQLWarning	0.000098
java.util.logging.Logger	0.000097
java.security.Security	0.000097
java.lang.Void	0.000096
java.text.DateFormatSymbols	0.000095
java.security.Guard	0.000095
java.util.EmptyStackException	0.000094
java.text.ParseException	0.000092
java.net.URISyntaxException	0.000090
java.util.PropertyResourceBundle	0.000090
java.security.CodeSource	0.000089
java.util.LinkedHashMap	0.000089
javax.security.auth.Subject	0.000088
java.util.logging.Level	0.000088
java.beans.PropertyChangeSupport	0.000087
java.lang.ref.FinalReference	0.000087
java.text.FieldPosition	0.000086
java.security.DomainCombiner	0.000085
javax.security.auth.login.LoginException	0.000085
java.util.FormatFlagsConversionMismatchException	0.000084
java.util.UnknownFormatConversionException	0.000084
java.util.DuplicateFormatFlagsException	0.000084
java.util.UnknownFormatFlagsException	0.000084
java.util.IllegalFormatConversionException	0.000084
java.util.IllegalFormatWidthException	0.000084
java.util.MissingFormatArgumentException	0.000084
java.util.IllegalFormatFlagsException	0.000084
java.util.FormatterClosedException	0.000084

Class Name	PageRank Score
java.util.Formattable	0.000084
java.util.IllegalFormatPrecisionException	0.000084
java.util.MissingFormatWidthException	0.000084
java.util.IllegalFormatCodePointException	0.000084
java.security.cert.CertificateException	0.000083
java.io.FilePermission	0.000080
java.net.SocketPermission	0.000080
java.security.SecurityPermission	0.000079
java.lang.NoSuchMethodError	0.000079
java.lang.reflect.InvocationHandler	0.000078
javax.security.auth.callback.Callback	0.000078
java.io.OptionalDataException	0.000077
java.nio.channels.ClosedChannelException	0.000074
java.net.NetPermission	0.000074
java.net.InetSocketAddress	0.000071
java.io.InputStreamReader	0.000070
java.io.BufferedReader	0.000070
java.util.logging.LogRecord	0.000070
java.lang.annotation.AnnotationFormatError	0.000069
java.text.ParsePosition	0.000068
java.net.SocketAddress	0.000068
java.sql.Connection	0.000068
java.lang.reflect.ReflectPermission	0.000067
java.security.spec.KeySpec	0.000067
java.nio.channels.SelectableChannel	0.000066
java.awt.font.FontRenderContext	0.000065
java.io.ObjectInputValidation	0.000065
java.security.PublicKey	0.000065
java.nio.IntBuffer	0.000065
java.nio.ShortBuffer	0.000065
java.nio.LongBuffer	0.000065
java.nio.FloatBuffer	0.000064
java.nio.DoubleBuffer	0.000064
java.util.WeakHashMap	0.000064
java.io.PushbackInputStream	0.000063
java.beans.FeatureDescriptor	0.000063
java.security.spec.AlgorithmParameterSpec	0.000062
java.lang.ArithmeticException	0.000062
java.io.StringWriter	0.000061
java.io.FilenameFilter	0.000061
java.beans.PropertyVetoException	0.000060
java.net.URLStreamHandler	0.000060

Class Name	PageRank Score
java.lang.IllegalAccessError	0.000060
java.lang.Readable	0.000059
java.io.WriteAbortedException	0.000059
java.net.URLConnection	0.000059
java.text.Collator	0.000059
java.net.SocketImpl	0.000059
java.sql.ResultSet	0.000057
java.security.cert.CertPath	0.000057
java.security.interfaces.DSAParams	0.000056
java.util.prefs.Preferences	0.000056
java.awt.font.TextHitInfo	0.000056
java.util.Observable	0.000056
java.text.AttributedString	0.000055
java.lang.UnsatisfiedLinkError	0.000055
java.util.TreeMap	0.000054
java.security.spec.ECParameterSpec	0.000053
java.sql.Array	0.000053
java.io.FileSystem	0.000053
java.nio.channels.spi.AbstractSelector	0.000053
java.security.PrivateKey	0.000053
java.nio.channels.Selector	0.000052
java.io.FileFilter	0.000051
java.util.AbstractSequentialList	0.000051
java.nio.channels.SocketChannel	0.000051
javax.security.auth.DestroyFailedException	0.000051
java.net.Proxy	0.000051
java.text.DateFormat	0.000051
java.util.InvalidPropertiesFormatException	0.000050
java.util.XMLUtils	0.000050
java.security.Provider	0.000050
java.nio.channels.AsynchronousCloseException	0.000049
java.security.NoSuchProviderException	0.000049
java.io.StringReader	0.000048
java.util.GregorianCalendar	0.000048
javax.sql.RowSet	0.000048
java.nio.DirectByteBuffer	0.000048
java.text.StringCharacterIterator	0.000047
java.security.InvalidAlgorithmParameterException	0.000047
java.util.BitSet	0.000047
java.security.KeyException	0.000047
java.nio.channels.ReadableByteChannel	0.000046
javax.sql.RowSetInternal	0.000046

Class Name	PageRank Score
java.lang.OutOfMemoryError	0.000046
java.nio.channels.WritableByteChannel	0.000046
java.util.Queue	0.000046
javax.security.auth.login.AccountException	0.000045
java.util.SortedMap	0.000045
java.lang.annotation.ElementType	0.000045
java.net.NetworkInterface	0.000044
java.util.jar.Manifest	0.000044
javax.security.auth.callback.CallbackHandler	0.000044
java.nio.HeapByteBuffer	0.000043
java.beans.VetoableChangeListener	0.000043
java.nio.channels.spi.AbstractSelectableChannel	0.000043
java.net.URLStreamHandlerFactory	0.000043
java.util.jar.Attributes	0.000043
java.awt.font.GlyphJustificationInfo	0.000042
java.security.SecureRandom	0.000042
java.nio.charset.CoderMalfunctionError	0.000041
java.awt.font.TextAttribute	0.000041
java.lang.Process	0.000041
java.nio.HeapCharBuffer	0.000041
javax.sql.PooledConnection	0.000040
java.beans.BeanInfo	0.000040
java.text.RuleBasedBreakIterator	0.000040
java.nio.channels.GatheringByteChannel	0.000040
java.nio.channels.ScatteringByteChannel	0.000040
java.net.DatagramSocketImpl	0.000040
java.nio.channels.ServerSocketChannel	0.000040
java.security.acl.Permission	0.000040
java.nio.StringCharBuffer	0.000039
java.util.zip.ZipEntry	0.000039
java.text.DictionaryBasedBreakIterator	0.000039
javax.security.auth.RefreshFailedException	0.000039
java.sql.Time	0.000038
java.util.zip.Checksum	0.000038
java.security.acl.NotOwnerException	0.000038
javax.security.auth.callback.UnsupportedCallbackException	0.000038
java.nio.channels.DatagramChannel	0.000038
java.util.EventListenerProxy	0.000038
java.security.spec.EncodedKeySpec	0.000038
java.lang.NoClassDefFoundError	0.000037
java.text.EntryPair	0.000037
java.sql.Timestamp	0.000037

Class Name	PageRank Score
java.net.ContentHandler	0.000037
java.nio.channels.ByteChannel	0.000037
javax.security.auth.login.CredentialException	0.000036
java.lang.annotation.RetentionPolicy	0.000036
java.lang.AssertionStatusDirectives	0.000036
java.nio.MappedByteBuffer	0.000036
java.io.RandomAccessFile	0.000036
java.util.SortedSet	0.000036
java.io.CharArrayReader	0.000036
java.sql.Blob	0.000036
java.util.TooManyListenersException	0.000036
java.sql.Savepoint	0.000035
java.awt.font.LineMetrics	0.000035
java.util.logging.Handler	0.000035
java.lang.ProcessBuilder	0.000035
java.sql.Ref	0.000035
java.beans.Expression	0.000035
java.net.SocketOptions	0.000035
java.security.UnresolvedPermission	0.000035
java.security.cert.CRL	0.000035
java.util.TreeSet	0.000035
java.sql.Clob	0.000035
java.security.spec.ECPoint	0.000035
java.security.cert.CertificateEncodingException	0.000035
java.security.cert.CertificateFactory	0.000034
java.nio.channels.FileLock	0.000034
java.text.SimpleDateFormat	0.000034
java.security.UnresolvedPermissionCollection	0.000034
java.beans.BeanDescriptor	0.000034
java.net.Inet4Address	0.000034
java.util.zip.ZipInputStream	0.000034
java.util.zip.ZipException	0.000034
java.util.logging.LogManager	0.000034
java.nio.channels.Pipe	0.000034
java.beans.PropertyDescriptor	0.000034
java.util.zip.CRC32	0.000034
java.util.zip.Deflater	0.000034
java.beans.MethodDescriptor	0.000034
java.security.interfaces.RSAPrivateKey	0.000034
java.util.jar.JarFile	0.000033
java.util.jar.JarInputStream	0.000033
java.lang.ArrayStoreException	0.000033

Class Name	PageRank Score
java.sql.Date	0.000033
java.sql.ResultSetMetaData	0.000033
java.util.zip.DeflaterOutputStream	0.000033
java.util.zip.Adler32	0.000033
java.security.cert.X509Certificate	0.000033
java.nio.charset.UnmappableCharacterException	0.000033
javax.security.auth.x500.X500Principal	0.000033
java.beans.Statement	0.000033
java.beans.Introspector	0.000033
java.security.InvalidParameterException	0.000033
java.beans.NameGenerator	0.000032
java.security.CodeSigner	0.000032
java.util.Currency	0.000032
javax.sql.RowSetMetaData	0.000032
javax.sql.RowSetWriter	0.000032
java.text.ChoiceFormat	0.000032
java.lang.InheritableThreadLocal	0.000032
java.security.InvalidKeyException	0.000032
java.beans.EventSetDescriptor	0.000032
javax.sql.ConnectionEventListener	0.000031
java.security.interfaces.DSAKey	0.000031
java.security.spec.RSAPrivateKeySpec	0.000031
java.net.DatagramPacket	0.000031
java.security.interfaces.RSAKey	0.000031
java.awt.font.TextLayout	0.000031
java.util.zip.InflaterInputStream	0.000031
java.nio.HeapShortBuffer	0.000031
java.nio.HeapIntBuffer	0.000031
java.nio.HeapLongBuffer	0.000031
java.nio.HeapFloatBuffer	0.000031
java.nio.HeapDoubleBuffer	0.000031
java.util.zip.Inflater	0.000031
java.security.interfaces.ECKey	0.000031
java.sql.Statement	0.000031
java.security.acl.AclEntry	0.000030
java.util.jar.JarEntry	0.000030
java.awt.font.GlyphVector	0.000030
javax.security.auth.AuthPermission	0.000030
java.util.jar.JarException	0.000030
java.nio.ByteBufferAsShortBufferB	0.000030
java.nio.ByteBufferAsIntBufferB	0.000030
java.nio.ByteBufferAsIntBufferL	0.000030

Class Name	PageRank Score
java.nio.ByteBufferAsLongBufferL	0.000030
java.nio.ByteBufferAsLongBufferB	0.000030
java.nio.ByteBufferAsDoubleBufferB	0.000030
java.nio.ByteBufferAsDoubleBufferL	0.000030
java.nio.ByteBufferAsShortBufferL	0.000030
java.nio.ByteBufferAsFloatBufferB	0.000030
java.nio.ByteBufferAsFloatBufferL	0.000030
java.security.spec.ECField	0.000030
java.security.acl.LastOwnerException	0.000029
java.util.Observer	0.000029
java.security.SignatureException	0.000029
java.net.Inet6Address	0.000029
java.net.InetAddressImpl	0.000029
java.security.cert.CertPathParameters	0.000029
java.util.AbstractQueue	0.000029
java.security.cert.X509Extension	0.000029
java.awt.font.GraphicAttribute	0.000029
java.nio.ByteBufferAsShortBufferRB	0.000029
java.nio.ByteBufferAsIntBufferRL	0.000029
java.nio.ByteBufferAsIntBufferRB	0.000029
java.nio.ByteBufferAsDoubleBufferRL	0.000029
java.nio.ByteBufferAsDoubleBufferRB	0.000029
java.nio.ByteBufferAsFloatBufferRL	0.000029
java.nio.ByteBufferAsFloatBufferRB	0.000029
java.nio.ByteBufferAsShortBufferRL	0.000029
java.nio.ByteBufferAsLongBufferRL	0.000029
java.nio.ByteBufferAsLongBufferRB	0.000029
javax.sql.ConnectionEvent	0.000029
java.security.spec.EllipticCurve	0.000029
java.security.spec.RSAOtherPrimeInfo	0.000028
java.security.cert.TrustAnchor	0.000028
java.util.zip.GZIPOutputStream	0.000028
javax.sql.RowSetEvent	0.000028
java.text.RuleBasedCollator	0.000028
javax.security.auth.Destroyable	0.000028
java.text.RBCollationTables	0.000028
java.util.IdentityHashMap	0.000028
java.nio.DirectLongBufferS	0.000028
java.nio.DirectLongBufferU	0.000028
java.nio.DirectFloatBufferU	0.000028
java.nio.DirectIntBufferU	0.000028
java.nio.DirectIntBufferS	0.000028

Class Name	PageRank Score
java.nio.DirectShortBufferS	0.000028
java.nio.DirectShortBufferU	0.000028
java.nio.DirectDoubleBufferU	0.000028
javax.sql.RowSetReader	0.000028
java.lang.ref.Finalizer	0.000028
javax.security.auth.callback.TextInputCallback	0.000028
javax.security.auth.callback.ChoiceCallback	0.000028
java.security.cert.CertStoreParameters	0.000028
java.net.PlainSocketImpl	0.000028
java.beans.VetoableChangeSupport	0.000028
java.awt.font.CharArrayIterator	0.000028
java.beans.IntrospectionException	0.000028
java.security.AlgorithmParameters	0.000027
java.nio.DirectFloatBufferS	0.000027
java.nio.DirectDoubleBufferS	0.000027
java.nio.DirectIntBufferRU	0.000027
java.nio.DirectIntBufferRS	0.000027
java.nio.DirectFloatBufferRU	0.000027
java.nio.DirectDoubleBufferRU	0.000027
java.nio.DirectShortBufferRU	0.000027
java.nio.DirectShortBufferRS	0.000027
java.nio.DirectLongBufferRS	0.000027
java.nio.DirectLongBufferRU	0.000027
java.text.CollationKey	0.000027
java.beans.Encoder	0.000027
java.sql.PreparedStatement	0.000027
java.nio.charset.spi.CharsetProvider	0.000027
java.io.CharArrayWriter	0.000027
java.nio.HeapShortBufferR	0.000027
java.nio.HeapIntBufferR	0.000027
java.nio.HeapLongBufferR	0.000027
java.nio.HeapFloatBufferR	0.000027
java.nio.HeapDoubleBufferR	0.000027
java.nio.DirectDoubleBufferRS	0.000027
java.nio.DirectFloatBufferRS	0.000027
java.io.FilterReader	0.000027
java.beans.PersistenceDelegate	0.000027
java.text.PatternEntry	0.000027
java.util.zip.ZipOutputStream	0.000027
java.util.SimpleTimeZone	0.000026
java.util.zip.CheckedInputStream	0.000026
java.security.acl.Acl	0.000026

Class Name	PageRank Score
java.util.zip.GZIPInputStream	0.000026
java.nio.ByteBufferAsCharBufferL	0.000026
java.nio.ByteBufferAsCharBufferB	0.000026
java.net.SocksSocketImpl	0.000026
java.security.cert.CertSelector	0.000026
java.beans.ExceptionListener	0.000026
java.util.prefs.PreferenceChangeEvent	0.000026
java.util.prefs.NodeChangeEvent	0.000026
java.nio.channels.spi.AbstractSelectionKey	0.000026
java.util.logging.Formatter	0.000026
java.sql.CallableStatement	0.000026
java.sql.DatabaseMetaData	0.000026
java.security.cert.CertPathBuilderResult	0.000026
java.nio.channels.ClosedByInterruptException	0.000026
java.nio.channels.InterruptibleChannel	0.000026
java.security.DigestException	0.000026
java.security.MessageDigestSpi	0.000026
java.util.zip.DataFormatException	0.000026
java.util.logging.Filter	0.000026
java.util.zip.ZipConstants	0.000026
java.net.SocketImplFactory	0.000026
java.net.DatagramSocket	0.000026
java.io.Win32FileSystem	0.000025
java.security.cert.CRLException	0.000025
java.security.cert.CertPathValidatorException	0.000025
java.beans.Visibility	0.000025
java.beans.DesignMode	0.000025
java.nio.ByteBufferAsCharBufferRB	0.000025
java.nio.ByteBufferAsCharBufferRL	0.000025
java.text.CharacterIteratorFieldDelegate	0.000025
java.security.acl.Owner	0.000025
java.security.cert.CertPathValidatorResult	0.000025
java.util.prefs.PreferencesFactory	0.000025
java.text.CollationElementIterator	0.000025
java.io.SequenceInputStream	0.000025
java.io.PipedWriter	0.000025
java.io.PipedReader	0.000025
javax.security.auth.callback.PasswordCallback	0.000025
java.sql.SQLInput	0.000025
java.util.prefs.WindowsPreferences	0.000025
java.text.DontCareFieldPosition	0.000025
java.lang.annotation.AnnotationTypeMismatchException	0.000025

Class Name	PageRank Score
java.sql.SQLOutput	0.000025
java.nio.HeapCharBufferR	0.000025
java.security.Timestamp	0.000025
java.security.KeyPair	0.000024
java.awt.font.NumericShaper	0.000024
java.nio.DirectCharBufferS	0.000024
java.nio.DirectCharBufferU	0.000024
java.io.PipedInputStream	0.000024
java.io.PipedOutputStream	0.000024
java.awt.font.TransformAttribute	0.000024
java.net.CacheResponse	0.000024
java.security.spec.InvalidParameterSpecException	0.000024
java.security.AlgorithmParametersSpi	0.000024
java.text.RBTableBuilder	0.000024
java.security.cert.CertPathBuilderException	0.000024
java.util.Timer	0.000024
java.util.TimerTask	0.000024
javax.security.auth.SubjectDomainCombiner	0.000024
java.lang.ExceptionInInitializerError	0.000024
java.nio.DirectCharBufferRS	0.000024
java.nio.DirectCharBufferRU	0.000024
java.beans.PropertyEditor	0.000024
java.io.FilterWriter	0.000024
java.security.spec.InvalidKeySpecException	0.000024
java.util.jar.JarVerifier	0.000024
java.util.zip.ZipFile	0.000024
java.security.DigestOutputStream	0.000024
java.text.BreakDictionary	0.000024
java.security.KeyStoreException	0.000024
java.security.KeyStore	0.000024
java.security.cert.CRLSelector	0.000024
java.text.MergeCollation	0.000024
java.util.EnumSet	0.000024
javax.security.auth.callback.NameCallback	0.000024
java.text.DigitList	0.000024
javax.security.auth.login.FailedLoginException	0.000024
javax.sql.XAConnection	0.000023
java.util.logging.StreamHandler	0.000023
java.io.ExpiringCache	0.000023
java.sql.DriverPropertyInfo	0.000023
java.beans.ReflectionUtils	0.000023
java.text.Annotation	0.000023

Class Name	PageRank Score
java.util.jar.JarOutputStream	0.000023
java.sql.SQLPermission	0.000023
java.sql.Driver	0.000023
java.security.Identity	0.000023
java.security.IdentityScope	0.000023
java.sql.Struct	0.000023
java.util.prefs.BackingStoreException	0.000023
java.util.logging.SimpleFormatter	0.000023
java.util.prefs.Base64	0.000023
java.text.Bidi	0.000023
java.nio.channels.IllegalBlockingModeException	0.000023
java.lang.Compiler	0.000023
java.util.logging.ErrorManager	0.000023
java.security.cert.X509CRLSelector	0.000023
java.security.KeyManagementException	0.000023
java.net.HttpURLConnection	0.000023
java.security.cert.CertStoreException	0.000023
java.security.cert.PolicyNode	0.000023
java.security.acl.Group	0.000023
java.lang.IllegalMonitorStateException	0.000023
java.net.Inet4AddressImpl	0.000023
java.io.StreamTokenizer	0.000023
java.net.SocketInputStream	0.000023
java.security.cert.X509CertSelector	0.000023
java.io.FileReader	0.000023
java.sql.SQLData	0.000023
java.io.LineNumberReader	0.000023
java.security.cert.PolicyQualifierInfo	0.000023
java.security.KeyFactorySpi	0.000022
java.beans.PropertyChangeListenerProxy	0.000022
java.beans.IndexedPropertyChangeEvent	0.000022
java.security.cert.X509CRLEntry	0.000022
javax.security.auth.PrivateCredentialPermission	0.000022
java.lang.ProcessImpl	0.000022
java.io.FileWriter	0.000022
java.awt.font.StyledParagraph	0.000022
javax.sql.RowSetListener	0.000022
java.text.CollationRules	0.000022
java.net.SocketOutputStream	0.000022
java.util.logging.XMLFormatter	0.000022
java.util.RegularEnumSet	0.000022
java.net.URLEncoder	0.000022

Class Name	PageRank Score
java.net.Authenticator	0.000022
java.io.PushbackReader	0.000022
java.security.cert.CertPathHelperImpl	0.000022
java.io.LineNumberInputStream	0.000022
java.util.PriorityQueue	0.000022
java.util.JumboEnumSet	0.000022
java.beans.ParameterDescriptor	0.000022
java.security.cert.CertificateNotYetValidException	0.000022
java.security.cert.CertificateExpiredException	0.000022
java.security.SecureRandomSpi	0.000022
java.beans.MetaData	0.000022
java.security.cert.PKIXParameters	0.000022
java.util.prefs.XmlSupport	0.000022
java.net.PasswordAuthentication	0.000022
java.security.spec.ECFieldFp	0.000022
java.security.spec.ECFieldF2m	0.000022
java.net.URLDecoder	0.000022
java.net.ProxySelector	0.000022
java.security.spec.MGF1ParameterSpec	0.000022
java.security.KeyFactory	0.000022
java.nio.HeapByteBufferR	0.000021
java.util.prefs.AbstractPreferences	0.000021
javax.security.auth.login.Configuration	0.000021
javax.security.auth.login.AppConfigurationEntry	0.000021
java.awt.font.TextMeasurer	0.000021
java.util.logging.LoggingMXBean	0.000021
java.security.cert.CertificateParsingException	0.000021
java.security.cert.LDAPCertStoreParameters	0.000021
java.awt.font.ImageGraphicAttribute	0.000021
java.security.DigestInputStream	0.000021
java.beans.IndexedPropertyDescriptor	0.000021
javax.security.auth.callback.TextOutputCallback	0.000021
javax.security.auth.callback.ConfirmationCallback	0.000021
java.util.prefs.InvalidPreferencesFormatException	0.000021
java.awt.font.LineBreakMeasurer	0.000021
java.net.JarURLConnection	0.000021
java.beans.XMLDecoder	0.000021
java.util.prefs.PreferenceChangeListener	0.000021
java.util.prefs.NodeChangeListener	0.000021
java.awt.font.TextLine	0.000021
java.security.cert.X509CRL	0.000021
java.security.cert.PKIXCertPathValidatorResult	0.000021

Class Name	PageRank Score
java.net.PlainDatagramSocketImpl	0.000021
javax.security.auth.Policy	0.000021
java.net.UnknownServiceException	0.000021
java.io.StringBufferInputStream	0.000021
java.security.cert.CertStore	0.000021
java.security.SecureClassLoader	0.000021
java.awt.font.GlyphMetrics	0.000021
java.awt.font.ShapeGraphicAttribute	0.000021
java.beans.SimpleBeanInfo	0.000021
java.security.cert.CertificateFactorySpi	0.000021
java.security.UnrecoverableKeyException	0.000021
java.net.CacheRequest	0.000021
java.net.FileNameMap	0.000021
java.net.ContentHandlerFactory	0.000021
java.net.Inet6AddressImpl	0.000021
java.beans.PropertyEditorSupport	0.000021
java.util.logging.ConsoleHandler	0.000021
java.lang.UnknownError	0.000021
java.net.URLClassLoader	0.000021
java.security.Signature	0.000021
java.nio.DirectByteBufferR	0.000021
java.security.AlgorithmParameterGenerator	0.000021
java.security.AlgorithmParameterGeneratorSpi	0.000021
java.sql.Types	0.000021
java.security.Certificate	0.000021
java.beans.EventHandler	0.000020
java.security.spec.X509EncodedKeySpec	0.000020
java.security.spec.PKCS8EncodedKeySpec	0.000020
java.security.SignatureSpi	0.000020
java.security.ProviderException	0.000020
java.util.jar.JavaUtilJarAccessImpl	0.000020
java.sql.ParameterMetaData	0.000020
java.security.UnrecoverableEntryException	0.000020
java.security.KeyStoreSpi	0.000020
java.security.cert.CertPathBuilder	0.000020
java.security.cert.CertPathBuilderSpi	0.000020
java.util.logging.MemoryHandler	0.000020
java.security.Signer	0.000020
java.security.KeyPairGeneratorSpi	0.000020
java.security.cert.CertPathValidator	0.000020
java.security.KeyPairGenerator	0.000020
java.security.cert.CertPathValidatorSpi	0.000020

Class Name	PageRank Score
java.net.ProtocolException	0.000020
java.util.UUID	0.000020
java.security.SignedObject	0.000020
java.security.cert.CertStoreSpi	0.000020
java.beans.VetoableChangeListenerProxy	0.000020
java.nio.channels.Channels	0.000020
java.beans.PropertyEditorManager	0.000020
javax.security.auth.login.LoginContext	0.000020
java.util.logging.SocketHandler	0.000020
javax.security.auth.Refreshable	0.000020
java.beans.DefaultPersistenceDelegate	0.000020
java.util.jar.Pack200	0.000020
java.beans.AppletInitializer	0.000020
java.beans.Beans	0.000020
java.net.MulticastSocket	0.000020
java.lang.InstantiationError	0.000020
java.security.cert.PKIXCertPathChecker	0.000020
java.net.DatagramSocketImplFactory	0.000020
java.beans.XMLEncoder	0.000020
java.util.logging.FileHandler	0.000020
java.util.EnumMap	0.000020
java.util.logging.LoggingPermission	0.000020
java.util.logging.Logging	0.000020
java.lang.reflect.UndeclaredThrowableException	0.000020
java.net.SocksConsts	0.000020
java.awt.font.TextJustifier	0.000020
java.util.InputMismatchException	0.000020
java.util.Scanner	0.000020
javax.security.auth.x500.X500PrivateCredential	0.000019
java.security.AuthProvider	0.000019
java.sql.DataTruncation	0.000019
java.nio.channels.NotYetConnectedException	0.000019
java.security.cert.CollectionCertStoreParameters	0.000019
java.nio.channels.ClosedSelectorException	0.000019
javax.security.auth.login.AccountLockedException	0.000019
java.net.HttpRetryException	0.000019
java.lang.annotation.Target	0.000019
java.nio.channels.IllegalSelectorException	0.000019
java.security.interfaces.DSAPrivateKey	0.000019
java.security.spec.DSAPublicKeySpec	0.000019
java.security.spec.DSAParameterSpec	0.000019
java.nio.channels.NotYetBoundException	0.000019

Class Name	PageRank Score
java.security.spec.DSAPrivateKeySpec	0.000019
java.beans.Customizer	0.000019
java.awt.font.OpenType	0.000019
java.nio.channels.NoConnectionPendingException	0.000019
java.sql.BatchUpdateException	0.000019
java.util.CurrencyData	0.000019
java.security.spec.PSSParameterSpec	0.000019
java.security.acl.AclNotFoundException	0.000019
java.security.spec.ECGenParameterSpec	0.000019
java.net.SocketTimeoutException	0.000019
java.nio.channels.UnresolvedAddressException	0.000019
java.util.zip.CheckedOutputStream	0.000019
java.security.interfaces.DSAPublicKey	0.000019
java.nio.channels.NonReadableChannelException	0.000019
java.security.spec.ECPrivateKeySpec	0.000019
javax.security.auth.callback.LanguageCallback	0.000019
java.awt.font.MultipleMaster	0.000019
javax.security.auth.login.AccountExpiredException	0.000019
java.lang.NoSuchFieldError	0.000019
java.lang.SuppressWarnings	0.000019
java.lang.reflect.ParameterizedType	0.000019
java.security.interfaces.RSAPublicKey	0.000019
java.lang.StackOverflowError	0.000019
java.lang.reflect.GenericSignatureFormatError	0.000019
java.nio.channels.AlreadyConnectedException	0.000019
java.security.cert.PKIXCertPathBuilderResult	0.000019
java.security.interfaces.DSAKeyPairGenerator	0.000019
java.lang.TypeNotPresentException	0.000019
java.nio.channels.OverlappingFileLockException	0.000019
java.security.spec.RSAPublicKeySpec	0.000019
java.net.ResponseCache	0.000019
java.lang.EnumConstantNotPresentException	0.000019
javax.security.auth.login.CredentialExpiredException	0.000019
java.nio.channels.CancelledKeyException	0.000019
javax.security.auth.login.AccountNotFoundException	0.000019
java.nio.channels.ConnectionPendingException	0.000019
java.net.ConnectException	0.000019
java.lang.VerifyError	0.000019
java.lang.UnsupportedClassVersionError	0.000019
java.security.spec.RSAKeyGenParameterSpec	0.000019
java.security.interfaces.RSAMultiPrimePrivateCrtKey	0.000019
java.lang.reflect.WildcardType	0.000019

Class Name	PageRank Score
java.net.SecureCacheResponse	0.000019
java.security.KeyRep	0.000019
javax.sql.XADataSource	0.000019
java.nio.channels.NonWritableChannelException	0.000019
javax.sql.DataSource	0.000019
java.security.interfaces.ECPublicKey	0.000019
java.security.cert.PKIXBuilderParameters	0.000019
java.nio.channels.FileLockInterruptionException	0.000019
javax.security.auth.login.CredentialNotFoundException	0.000019
java.security.spec.ECPublicKeySpec	0.000019
java.net.CookieHandler	0.000019
java.lang.ref.PhantomReference	0.000019
java.lang.annotation.IncompleteAnnotationException	0.000019
java.io.WinNTFileSystem	0.000019
java.nio.channels.UnsupportedAddressTypeException	0.000019
java.security.interfaces.RSAPrivateCrtKey	0.000019
java.lang.ClassCircularityError	0.000019
java.security.GuardedObject	0.000019
javax.sql.ConnectionPoolDataSource	0.000019
java.net.BindException	0.000019
java.util.prefs.WindowsPreferencesFactory	0.000019
java.security.spec.RSAMultiPrimePrivateCrtKeySpec	0.000019
java.security.spec.RSAPrivateCrtKeySpec	0.000019
java.security.interfaces.ECPrivateKey	0.000019
java.net.NoRouteToHostException	0.000019
java.lang.annotation.Retention	0.000019
java.lang.reflect.MalformedParameterizedTypeException	0.000019
java.net.PortUnreachableException	0.000019
java.util.FormattableFlags	0.000019

APPENDIX D – Android PageRank Full Dataset[270]

PageRank scores for the copied classes in the Android 5.1.0 source code, ordered from the highest score to the lowest score.

Class Name	PageRank Score
java.lang.String	0.121342
java.lang.Object	0.077886
java.lang.Throwable	0.036248
java.lang.Exception	0.026851
java.lang.Class	0.024376
java.lang.Override	0.017777
java.lang.RuntimeException	0.016621
java.lang.annotation.RetentionPolicy	0.016203
java.lang.Enum	0.015651
java.lang.Integer	0.015368
java.lang.SuppressWarnings	0.014415
java.lang.annotation.ElementType	0.013468
java.io.IOException	0.012908
java.lang.Comparable	0.012653
java.io.Serializable	0.012192
java.lang.NullPointerException	0.011221
java.lang.System	0.010571
java.lang.CloneNotSupportedException	0.010216
java.lang.Number	0.009661
java.lang.Cloneable	0.009598
java.lang.IllegalArgumentException	0.009480
java.lang.InterruptedException	0.009279
java.lang.annotation.Retention	0.008523
java.lang.annotation.Target	0.007099
java.lang.StringBuilder	0.006866
java.lang.CharSequence	0.006585
java.lang.IndexOutOfBoundsException	0.006368
java.lang.Character	0.005475
java.lang.Math	0.005252
java.lang.Float	0.005228
java.lang.Long	0.005170
java.lang.Double	0.005151
java.nio.charset.Charset	0.004951
java.io.BufferedInputStream	0.004945

[270] For the 61 class sensitivity analysis, the PageRank scores of the 61 classes are removed from this dataset for further analysis; for the java.lang sensitivity analysis, the PageRanks scores of all the classes in java.lang package are removed from this dataset for further analysis.

java.lang.ArrayIndexOutOfBoundsException	0.004722
java.util.Locale	0.004664
java.lang.Deprecated	0.004617
java.lang.AbstractStringBuilder	0.004572
java.lang.StringBuffer	0.004502
java.lang.IllegalStateException	0.004278
java.util.Arrays	0.004210
java.util.regex.Pattern	0.004138
java.util.regex.Matcher	0.004113
java.nio.CharBuffer	0.003844
java.nio.Buffer	0.003837
java.io.UnsupportedEncodingException	0.003765
java.util.regex.Splitter	0.003760
java.nio.ByteBuffer	0.003747
java.util.Comparator	0.003566
java.lang.CaseMapper	0.003565
java.util.Formatter	0.003483
java.lang.StringIndexOutOfBoundsException	0.003464
java.io.ObjectOutputStream	0.003314
java.nio.charset.Charsets	0.003138
java.io.PrintStream	0.003136
java.util.Iterator	0.003128
java.io.InputStream	0.002846
java.lang.StackTraceElement	0.002765
java.io.PrintWriter	0.002680
java.util.Collection	0.002386
java.lang.ClassNotFoundException	0.002145
java.lang.reflect.Method	0.001919
java.security.AccessController	0.001808
java.util.List	0.001782
java.util.ArrayList	0.001752
java.lang.reflect.AccessibleObject	0.001720
java.util.HashMap	0.001649
java.util.Set	0.001583
java.lang.reflect.TypeVariable	0.001554
java.lang.SecurityManager	0.001499
java.lang.annotation.Documented	0.001451
java.lang.reflect.Type	0.001425
java.io.ObjectInputStream	0.001403
java.lang.SecurityException	0.001301
java.lang.ClassLoader	0.001296
java.util.Map	0.001164
java.lang.ref.Reference	0.001157

java.nio.InvalidMarkException	0.001143
java.io.OutputStream	0.001115
java.io.File	0.001079
java.io.Closeable	0.001063
java.lang.IllegalAccessException	0.000998
java.lang.RuntimePermission	0.000994
java.lang.Error	0.000992
java.security.PrivilegedExceptionAction	0.000965
java.lang.reflect.Modifier	0.000965
java.util.Random	0.000961
java.lang.reflect.GenericDeclaration	0.000960
java.lang.ClassCastException	0.000932
java.lang.Appendable	0.000932
java.lang.annotation.Annotation	0.000904
java.lang.Thread	0.000887
java.lang.UnsupportedOperationException	0.000877
java.lang.NoSuchMethodException	0.000839
java.io.FileOutputStream	0.000816
java.lang.AssertionError	0.000805
java.net.URL	0.000786
java.lang.Iterable	0.000785
java.io.FileDescriptor	0.000785
java.util.Collections	0.000734
java.security.ProtectionDomain	0.000720
java.io.FilterInputStream	0.000717
java.lang.reflect.Constructor	0.000717
java.lang.IntegralToString	0.000706
java.lang.Boolean	0.000672
java.lang.reflect.Field	0.000669
java.lang.reflect.Member	0.000654
java.security.BasicPermission	0.000649
java.lang.InstantiationException	0.000621
java.io.ObjectStreamField	0.000618
java.lang.reflect.AnnotatedElement	0.000591
java.io.FileNotFoundException	0.000585
java.util.PropertyPermission	0.000570
java.lang.Package	0.000569
java.lang.RealToString	0.000564
java.lang.ref.ReferenceQueue	0.000550
java.lang.NoSuchFieldException	0.000535
java.util.Hashtable	0.000532
java.nio.ByteOrder	0.000521
java.nio.BufferOverflowException	0.000517

java.io.ObjectStreamException	0.000516
java.lang.Runnable	0.000504
java.lang.ClassCache	0.000504
java.lang.ref.SoftReference	0.000503
java.nio.BufferUnderflowException	0.000502
java.io.FileInputStream	0.000501
java.lang.ExceptionInInitializerError	0.000489
java.io.InvalidObjectException	0.000488
java.lang.annotation.Inherited	0.000487
java.util.ListIterator	0.000486
java.lang.Byte	0.000477
java.lang.Short	0.000470
java.lang.NegativeArraySizeException	0.000462
java.util.AbstractMap	0.000458
java.nio.BufferFactory	0.000452
java.security.PrivilegedAction	0.000437
java.security.Permission	0.000426
java.util.Properties	0.000420
java.util.IllegalFormatException	0.000416
java.lang.reflect.InvocationTargetException	0.000413
java.text.ParseException	0.000390
java.util.Enumeration	0.000387
java.nio.channels.spi.SelectorProvider	0.000385
java.nio.channels.Channel	0.000381
java.util.HashSet	0.000379
java.lang.Runtime	0.000376
java.io.FilterOutputStream	0.000354
java.io.Console	0.000344
java.util.regex.MatchResult	0.000325
java.util.regex.MatchResultImpl	0.000323
java.lang.LinkageError	0.000317
java.io.ByteArrayOutputStream	0.000316
java.io.Writer	0.000315
java.security.GeneralSecurityException	0.000311
java.lang.reflect.Array	0.000311
java.util.NoSuchElementException	0.000308
java.lang.ThreadLocal	0.000295
java.nio.IntBuffer	0.000295
java.util.Vector	0.000289
java.nio.charset.IllegalCharsetNameException	0.000288
java.nio.FloatBuffer	0.000273
java.util.AbstractList	0.000272
java.lang.NumberFormatException	0.000261

java.util.RandomAccess	0.000259
java.io.Flushable	0.000254
java.nio.ShortBuffer	0.000253
java.io.OutputStreamWriter	0.000253
java.util.Date	0.000250
java.util.AbstractCollection	0.000245
java.util.regex.PatternSyntaxException	0.000245
java.security.PermissionCollection	0.000240
java.util.MissingResourceException	0.000237
java.nio.LongBuffer	0.000228
java.nio.DoubleBuffer	0.000227
java.nio.charset.CodingErrorAction	0.000226
java.nio.charset.CharsetDecoder	0.000220
java.nio.charset.CharacterCodingException	0.000220
java.nio.charset.CharsetEncoder	0.000220
java.lang.Readable	0.000217
java.io.Reader	0.000210
java.security.Key	0.000207
java.io.StringWriter	0.000197
java.net.InetAddress	0.000197
java.util.ConcurrentModificationException	0.000184
java.io.ByteArrayInputStream	0.000183
java.util.TreeMap	0.000180
java.io.BufferedOutputStream	0.000177
java.util.SortedMap	0.000177
java.security.AccessControlContext	0.000175
java.util.ServiceLoader	0.000174
java.util.AbstractSet	0.000169
java.security.Security	0.000164
java.util.Calendar	0.000164
java.sql.SQLException	0.000163
java.nio.charset.UnsupportedCharsetException	0.000159
java.lang.IncompatibleClassChangeError	0.000156
java.util.ComparableTimSort	0.000153
java.util.DualPivotQuicksort	0.000153
java.io.SyncFailedException	0.000153
java.util.TimSort	0.000151
java.util.TimeZone	0.000150
java.nio.charset.spi.CharsetProvider	0.000147
java.lang.ArrayStoreException	0.000147
java.security.Principal	0.000138
java.util.WeakHashMap	0.000137
java.lang.ref.WeakReference	0.000136

java.security.Policy	0.000136
java.security.NoSuchAlgorithmException	0.000131
java.io.DataOutputStream	0.000126
java.security.SecureRandom	0.000122
java.util.StringTokenizer	0.000121
java.net.URI	0.000121
java.io.InputStreamReader	0.000114
java.util.ListResourceBundle	0.000113
java.net.MalformedURLException	0.000109
java.nio.channels.FileChannel	0.000102
java.net.URISyntaxException	0.000101
java.io.EmulatedFields	0.000100
java.security.AccessControlException	0.000100
java.lang.NoSuchFieldError	0.000100
java.security.PrivilegedActionException	0.000100
java.security.DomainCombiner	0.000099
java.io.BufferedReader	0.000099
java.io.ObjectStreamClass	0.000099
java.lang.reflect.Proxy	0.000097
java.io.StreamCorruptedException	0.000095
java.nio.channels.spi.AbstractInterruptibleChannel	0.000095
java.lang.reflect.ParameterizedType	0.000094
java.lang.ClassFormatError	0.000094
java.io.SerializablePermission	0.000094
java.security.cert.Certificate	0.000094
java.io.ObjectStreamConstants	0.000092
java.io.NotActiveException	0.000092
java.io.DataInputStream	0.000092
java.io.ObjectOutput	0.000090
java.io.Externalizable	0.000089
java.util.ResourceBundle	0.000089
java.io.InvalidClassException	0.000088
java.io.BufferedWriter	0.000087
java.io.NotSerializableException	0.000085
java.lang.InternalError	0.000085
java.security.Provider	0.000083
java.net.SocketException	0.000082
java.security.spec.AlgorithmParameterSpec	0.000082
java.lang.reflect.GenericArrayType	0.000081
java.security.PublicKey	0.000079
java.lang.ThreadGroup	0.000079
java.security.CodeSource	0.000079
java.security.cert.CertificateException	0.000078

java.net.Socket	0.000078
java.lang.ThreadDeath	0.000076
java.lang.VirtualMachineError	0.000075
java.lang.ArithmeticException	0.000075
java.lang.reflect.ReflectPermission	0.000075
java.lang.reflect.ReflectionAccessImpl	0.000074
java.text.DateFormat	0.000074
java.io.DataOutput	0.000074
java.util.IdentityHashMap	0.000072
java.util.LinkedList	0.000070
java.io.EmulatedFieldsForDumping	0.000068
java.util.EventObject	0.000068
java.security.SecurityPermission	0.000067
java.security.InvalidAlgorithmParameterException	0.000066
java.security.PrivateKey	0.000065
java.nio.ReadOnlyBufferException	0.000062
java.net.UnknownHostException	0.000061
java.lang.VerifyError	0.000060
java.util.FormatterClosedException	0.000060
javax.net.ssl.SSLSession	0.000059
java.security.InvalidKeyException	0.000059
java.util.IllegalFormatFlagsException	0.000059
java.lang.Void	0.000059
java.util.EventListener	0.000059
java.util.Formattable	0.000059
java.util.IllegalFormatCodePointException	0.000059
java.util.FormattableFlags	0.000059
java.util.IllegalFormatPrecisionException	0.000059
java.util.IllegalFormatWidthException	0.000059
java.util.FormatFlagsConversionMismatchException	0.000059
java.util.DuplicateFormatFlagsException	0.000059
java.util.MissingFormatArgumentException	0.000059
java.util.MissingFormatWidthException	0.000059
java.util.IllegalFormatConversionException	0.000058
java.util.UnknownFormatConversionException	0.000058
java.security.NoSuchProviderException	0.000057
java.nio.charset.CoderResult	0.000056
java.lang.AbstractMethodError	0.000056
java.net.InetSocketAddress	0.000056
javax.net.ssl.SSLException	0.000055
java.net.SocketPermission	0.000055
java.security.spec.KeySpec	0.000055
java.io.FilePermission	0.000054

java.nio.channels.SocketChannel	0.000054
java.security.cert.X509Certificate	0.000054
java.io.FilenameFilter	0.000053
java.io.FileFilter	0.000052
java.security.AllPermission	0.000052
java.net.URLConnection	0.000051
java.security.BasicPermissionCollection	0.000051
java.lang.VMClassLoader	0.000050
java.security.Guard	0.000050
java.nio.channels.SelectionKey	0.000049
java.text.SimpleDateFormat	0.000048
java.security.KeyException	0.000048
java.lang.EmptyEnumeration	0.000047
java.security.cert.CertPath	0.000047
java.net.URLStreamHandler	0.000046
java.net.SocketAddress	0.000046
java.io.ObjectInput	0.000045
java.lang.InheritableThreadLocal	0.000045
java.util.GregorianCalendar	0.000045
java.util.Queue	0.000044
java.security.KeyStore	0.000044
java.lang.NoSuchMethodError	0.000044
java.nio.channels.ClosedChannelException	0.000044
java.util.logging.Logger	0.000043
java.security.InvalidParameterException	0.000043
java.util.SortedSet	0.000042
java.net.ServerSocket	0.000042
java.text.Collator	0.000041
java.util.logging.Level	0.000041
java.lang.reflect.GenericSignatureFormatError	0.000040
java.lang.IllegalThreadStateException	0.000040
java.util.PropertyPermissionCollection	0.000039
java.security.cert.CertificateEncodingException	0.000039
java.security.KeyPair	0.000039
java.sql.Connection	0.000039
java.io.StringReader	0.000039
java.nio.charset.MalformedInputException	0.000038
java.net.Proxy	0.000038
java.io.DataInput	0.000038
java.nio.channels.spi.AbstractSelector	0.000038
java.nio.charset.UnmappableCharacterException	0.000038
java.security.cert.CertificateFactory	0.000036
java.security.MessageDigest	0.000036

java.nio.channels.DatagramChannel	0.000036
java.nio.ReadWriteDirectByteBuffer	0.000036
java.nio.channels.ServerSocketChannel	0.000036
java.lang.OutOfMemoryError	0.000036
java.security.interfaces.DSAParams	0.000036
javax.security.auth.x500.X500Principal	0.000036
java.nio.channels.spi.AbstractSelectableChannel	0.000035
javax.crypto.SecretKey	0.000035
java.net.SocketImpl	0.000035
java.lang.VMThread	0.000035
java.text.NumberFormat	0.000034
java.nio.channels.ReadableByteChannel	0.000034
java.text.AttributedCharacterIterator	0.000034
java.io.EOFException	0.000034
java.nio.channels.Pipe	0.000034
java.nio.ReadWriteDoubleArrayBuffer	0.000033
java.nio.ReadWriteIntArrayBuffer	0.000033
java.nio.ReadWriteShortArrayBuffer	0.000033
java.nio.ReadWriteLongArrayBuffer	0.000033
java.nio.ReadWriteCharArrayBuffer	0.000033
java.nio.ReadWriteFloatArrayBuffer	0.000033
java.util.BitSet	0.000033
java.nio.channels.Selector	0.000033
java.util.TreeSet	0.000032
javax.crypto.Cipher	0.000032
java.util.Observable	0.000032
java.net.URLEncoder	0.000032
java.nio.ReadWriteHeapByteBuffer	0.000032
java.nio.CharSequenceAdapter	0.000032
java.util.zip.ZipEntry	0.000032
java.util.Stack	0.000032
java.lang.UnsatisfiedLinkError	0.000032
java.net.NetPermission	0.000031
java.util.EnumSet	0.000031
java.lang.Process	0.000031
java.text.Format	0.000030
java.security.SignatureException	0.000030
java.nio.channels.WritableByteChannel	0.000030
java.util.LinkedHashMap	0.000030
java.text.DateFormatSymbols	0.000030
java.util.prefs.Preferences	0.000030
java.io.FileReader	0.000029
java.io.OptionalDataException	0.000029

java.util.logging.LogRecord	0.000029
java.sql.Clob	0.000029
java.nio.charset.CoderMalfunctionError	0.000029
java.util.AbstractQueue	0.000029
java.text.MessageFormat	0.000029
java.security.KeyStoreException	0.000029
java.security.KeyPairGenerator	0.000029
java.security.AlgorithmParameters	0.000029
java.lang.TypeNotPresentException	0.000028
java.io.ObjectInputValidation	0.000028
java.lang.NoClassDefFoundError	0.000028
java.lang.IllegalAccessError	0.000028
java.sql.ResultSet	0.000028
java.nio.channels.SelectableChannel	0.000028
java.io.WriteAbortedException	0.000027
javax.net.ssl.SSLPeerUnverifiedException	0.000027
java.sql.SQLNonTransientException	0.000027
java.nio.channels.AsynchronousCloseException	0.000027
javax.crypto.spec.DHParameterSpec	0.000027
java.io.IOError	0.000027
java.io.EmulatedFieldsForLoading	0.000027
java.util.Deque	0.000026
java.net.URLStreamHandlerFactory	0.000026
javax.security.cert.CertificateException	0.000026
java.net.Inet4Address	0.000026
java.util.MapEntry	0.000026
java.io.RandomAccessFile	0.000025
java.io.InterruptedIOException	0.000025
java.io.FileWriter	0.000025
java.security.spec.EncodedKeySpec	0.000025
java.security.KeyManagementException	0.000025
java.text.ParsePosition	0.000024
java.util.SimpleTimeZone	0.000024
javax.net.ssl.SSLSocketFactory	0.000024
java.nio.channels.GatheringByteChannel	0.000024
java.nio.channels.ScatteringByteChannel	0.000024
java.security.SecureRandomSpi	0.000024
java.io.PipedInputStream	0.000024
java.sql.Wrapper	0.000024
java.io.PipedOutputStream	0.000024
java.net.DatagramPacket	0.000023
java.util.Dictionary	0.000023
javax.net.ssl.SSLSessionContext	0.000023

java.text.CharacterIterator	0.000023
java.text.FieldPosition	0.000023
javax.net.ssl.TrustManager	0.000023
java.security.cert.CRL	0.000022
java.io.UTFDataFormatException	0.000022
java.security.cert.CertPathParameters	0.000022
java.security.spec.ECParameterSpec	0.000022
javax.security.cert.X509Certificate	0.000022
javax.net.ssl.SSLSocket	0.000022
javax.net.ssl.SSLParameters	0.000021
javax.net.ssl.ManagerFactoryParameters	0.000021
java.net.NetworkInterface	0.000021
java.security.spec.InvalidKeySpecException	0.000021
java.util.EmptyStackException	0.000021
java.nio.channels.ByteChannel	0.000021
java.io.PushbackInputStream	0.000021
java.nio.channels.FileLock	0.000021
java.util.Timer	0.000020
java.lang.IllegalMonitorStateException	0.000020
java.util.UUID	0.000020
java.net.SocketOptions	0.000020
java.util.zip.ZipFile	0.000020
java.util.zip.ZipException	0.000020
javax.net.ssl.KeyManager	0.000020
java.util.zip.Checksum	0.000020
java.text.DecimalFormatSymbols	0.000020
java.util.zip.Adler32	0.000020
java.nio.MappedByteBuffer	0.000020
java.util.TimerTask	0.000020
java.net.DatagramSocket	0.000020
java.text.DecimalFormat	0.000020
java.security.interfaces.RSAKey	0.000020
java.util.jar.Manifest	0.000019
java.sql.Statement	0.000019
java.security.cert.CRLException	0.000019
java.util.zip.CRC32	0.000019
java.security.UnrecoverableEntryException	0.000019
javax.security.auth.callback.Callback	0.000019
java.util.NavigableSet	0.000019
javax.crypto.ShortBufferException	0.000018
javax.crypto.spec.IvParameterSpec	0.000018
javax.security.auth.DestroyFailedException	0.000018
java.security.Signature	0.000018

java.sql.SQLWarning	0.000018
java.util.zip.ZipOutputStream	0.000018
javax.sql.PooledConnection	0.000018
java.util.zip.Deflater	0.000018
java.security.interfaces.DSAKey	0.000017
java.security.cert.CertPathValidatorException	0.000017
java.lang.ProcessManager	0.000017
java.security.spec.RSAPrivateKeySpec	0.000017
java.util.jar.Attributes	0.000017
java.sql.Date	0.000017
java.sql.Time	0.000017
java.security.UnrecoverableKeyException	0.000017
java.util.logging.LogManager	0.000017
java.security.cert.X509Extension	0.000017
java.security.AllPermissionCollection	0.000017
java.beans.PropertyChangeEvent	0.000017
java.security.Permissions	0.000017
java.security.KeyFactory	0.000017
java.sql.DriverManager	0.000017
javax.crypto.IllegalBlockSizeException	0.000017
java.security.interfaces.RSAPrivateKey	0.000017
java.net.DatagramSocketImpl	0.000017
java.security.spec.DSAParameterSpec	0.000017
java.lang.reflect.InvocationHandler	0.000017
java.sql.SQLTransientException	0.000017
javax.crypto.BadPaddingException	0.000017
javax.crypto.NoSuchPaddingException	0.000017
javax.net.ssl.SSLEngine	0.000017
java.sql.PreparedStatement	0.000017
javax.net.ssl.SSLServerSocketFactory	0.000017
javax.net.SocketFactory	0.000017
java.util.InvalidPropertiesFormatException	0.000016
javax.sql.RowSetInternal	0.000016
java.sql.Blob	0.000016
java.security.cert.CertStoreParameters	0.000016
java.util.AbstractSequentialList	0.000016
java.util.zip.Inflater	0.000016
java.nio.DirectByteBuffer	0.000016
java.security.spec.ECPoint	0.000016
java.security.acl.NotOwnerException	0.000016
java.sql.Array	0.000016
javax.net.ssl.SSLContext	0.000016
java.net.Inet6Address	0.000016

java.util.jar.JarEntry	0.000016
java.util.zip.DeflaterOutputStream	0.000016
java.net.ContentHandler	0.000016
javax.net.ssl.HandshakeCompletedEvent	0.000016
java.security.MessageDigestSpi	0.000016
java.util.Currency	0.000016
javax.sql.StatementEvent	0.000016
java.util.jar.JarFile	0.000015
java.text.CollationKey	0.000015
java.security.acl.Permission	0.000015
java.util.NavigableMap	0.000015
java.util.logging.Handler	0.000015
javax.sql.ConnectionEvent	0.000015
javax.crypto.interfaces.DHKey	0.000015
java.security.IdentityScope	0.000015
java.util.prefs.NodeChangeEvent	0.000015
java.util.zip.ZipInputStream	0.000015
java.sql.ResultSetMetaData	0.000015
java.text.AttributedString	0.000015
java.security.cert.CertPathBuilderException	0.000015
java.security.DigestException	0.000015
javax.crypto.spec.SecretKeySpec	0.000015
java.util.zip.DataFormatException	0.000015
java.util.Observer	0.000015
java.nio.DoubleArrayBuffer	0.000014
java.nio.FloatArrayBuffer	0.000014
java.nio.IntArrayBuffer	0.000014
java.nio.LongArrayBuffer	0.000014
java.nio.ShortArrayBuffer	0.000014
java.sql.Timestamp	0.000014
java.security.cert.CertPathBuilderResult	0.000014
java.util.prefs.PreferenceChangeEvent	0.000014
java.sql.Driver	0.000014
java.security.cert.TrustAnchor	0.000014
java.sql.SQLXML	0.000014
java.sql.NClob	0.000014
java.security.spec.RSAOtherPrimeInfo	0.000014
java.net.AddressCache	0.000014
java.security.acl.LastOwnerException	0.000014
java.sql.DatabaseMetaData	0.000014
javax.crypto.KeyGenerator	0.000014
java.security.spec.EllipticCurve	0.000014
javax.sql.ConnectionEventListener	0.000014

java.security.cert.CertificateParsingException	0.000014
java.security.cert.CertificateExpiredException	0.000014
java.security.cert.CertificateNotYetValidException	0.000014
java.nio.channels.ClosedByInterruptException	0.000014
java.sql.Ref	0.000014
java.net.HttpCookie	0.000013
java.nio.CharArrayBuffer	0.000013
java.security.cert.CertSelector	0.000013
java.security.Identity	0.000013
java.nio.ReadOnlyDirectByteBuffer	0.000013
java.util.zip.GZIPInputStream	0.000013
java.security.spec.InvalidParameterSpecException	0.000013
javax.crypto.spec.PSource	0.000013
javax.security.auth.callback.UnsupportedCallbackException	0.000013
javax.net.ServerSocketFactory	0.000013
java.security.cert.PKIXParameters	0.000013
java.util.zip.InflaterInputStream	0.000013
java.security.cert.CertPathValidatorResult	0.000013
java.security.spec.PKCS8EncodedKeySpec	0.000013
java.sql.DriverPropertyInfo	0.000013
java.nio.ReadOnlyShortArrayBuffer	0.000013
java.nio.ReadOnlyLongArrayBuffer	0.000013
java.nio.ReadOnlyIntArrayBuffer	0.000013
java.nio.ReadOnlyDoubleArrayBuffer	0.000013
java.nio.ReadOnlyFloatArrayBuffer	0.000013
java.nio.charset.ModifiedUtf8	0.000013
java.security.spec.MGF1ParameterSpec	0.000013
javax.net.ssl.SSLEngineResult	0.000013
javax.net.ssl.SSLSessionBindingEvent	0.000013
java.io.FilterReader	0.000013
javax.security.auth.callback.CallbackHandler	0.000013
java.security.CodeSigner	0.000013
java.security.spec.X509EncodedKeySpec	0.000013
java.util.logging.Formatter	0.000013
java.text.BreakIterator	0.000013
java.util.jar.JarOutputStream	0.000013
java.net.CacheRequest	0.000013
java.util.ServiceConfigurationError	0.000013
javax.sql.RowSetEvent	0.000012
java.lang.ref.PhantomReference	0.000012
java.util.Grego	0.000012
javax.net.ssl.HandshakeCompletedListener	0.000012
javax.net.ssl.SSLHandshakeException	0.000012

java.util.EnumMap	0.000012
javax.sql.StatementEventListener	0.000012
javax.sql.RowSetMetaData	0.000012
javax.net.ssl.KeyManagerFactory	0.000012
java.nio.ReadOnlyCharArrayBuffer	0.000012
javax.crypto.ExemptionMechanismException	0.000012
java.nio.HeapByteBuffer	0.000012
java.nio.channels.InterruptibleChannel	0.000012
java.net.URLDecoder	0.000012
javax.net.ssl.SSLServerSocket	0.000012
javax.sql.RowSet	0.000012
java.security.spec.ECField	0.000012
java.net.URLClassLoader	0.000012
java.text.RuleBasedCollator	0.000012
java.io.CharArrayReader	0.000012
java.net.SocketTimeoutException	0.000012
java.io.LineNumberReader	0.000012
java.util.prefs.BackingStoreException	0.000012
java.net.HttpURLConnection	0.000012
java.nio.channels.IllegalBlockingModeException	0.000012
java.lang.reflect.WildcardType	0.000012
java.io.FilterWriter	0.000012
java.sql.RowId	0.000012
javax.crypto.spec.PBEKeySpec	0.000012
java.net.BindException	0.000012
java.lang.ProcessBuilder	0.000012
java.net.SocketImplFactory	0.000011
java.net.ConnectException	0.000011
java.nio.channels.spi.AbstractSelectionKey	0.000011
java.security.interfaces.ECKey	0.000011
java.security.cert.CertStoreException	0.000011
javax.net.ssl.TrustManagerFactory	0.000011
java.net.Authenticator	0.000011
javax.net.ssl.X509TrustManager	0.000011
java.sql.Struct	0.000011
java.util.zip.GZIPOutputStream	0.000011
java.security.spec.RSAPublicKeySpec	0.000011
javax.security.auth.AuthPermission	0.000011
java.net.URIEncoderDecoder	0.000011
java.lang.reflect.MalformedParameterizedTypeException	0.000011
javax.crypto.NullCipher	0.000011
java.security.cert.CRLSelector	0.000011
java.util.prefs.NodeChangeListener	0.000011

java.security.interfaces.RSAPublicKey	0.000011
javax.net.ssl.HostnameVerifier	0.000011
java.net.PasswordAuthentication	0.000011
java.nio.ReadOnlyHeapByteBuffer	0.000011
java.sql.Savepoint	0.000011
java.security.interfaces.DSAPrivateKey	0.000011
java.security.interfaces.DSAPublicKey	0.000011
java.security.PolicySpi	0.000011
java.security.AlgorithmParameterGenerator	0.000011
java.util.prefs.PreferencesFactory	0.000011
java.security.cert.X509CertSelector	0.000011
javax.net.ssl.X509KeyManager	0.000011
java.sql.SQLInput	0.000011
java.nio.channels.CancelledKeyException	0.000011
java.util.zip.ZipConstants	0.000011
java.net.FileNameMap	0.000011
java.net.IDN	0.000011
java.security.acl.Group	0.000011
java.sql.BatchUpdateException	0.000011
java.util.LinkedHashSet	0.000011
java.security.cert.X509CRLEntry	0.000011
java.util.PropertyResourceBundle	0.000011
javax.security.auth.Subject	0.000011
java.util.prefs.PreferenceChangeListener	0.000011
javax.crypto.SecretKeyFactory	0.000011
java.security.spec.ECFieldF2m	0.000011
java.sql.CallableStatement	0.000011
java.security.cert.PolicyNode	0.000011
java.sql.Types	0.000010
javax.crypto.Mac	0.000010
java.io.CharArrayWriter	0.000010
java.util.logging.StreamHandler	0.000010
javax.crypto.spec.DESKeySpec	0.000010
java.security.cert.X509CRL	0.000010
java.io.PipedReader	0.000010
java.util.Scanner	0.000010
javax.sql.CommonDataSource	0.000010
javax.net.ssl.TrustManagerFactorySpi	0.000010
java.security.acl.AclEntry	0.000010
java.security.AlgorithmParametersSpi	0.000010
javax.crypto.CipherSpi	0.000010
javax.net.ssl.HttpsURLConnection	0.000010
java.security.spec.DSAPublicKeySpec	0.000010

java.text.Normalizer	0.000010
java.security.spec.ECFieldFp	0.000010
java.io.PushbackReader	0.000010
java.lang.InstantiationError	0.000010
java.sql.SQLOutput	0.000010
java.security.cert.CertificateFactorySpi	0.000010
java.util.logging.Filter	0.000010
java.nio.channels.IllegalSelectorException	0.000010
java.text.Annotation	0.000010
java.security.cert.CollectionCertStoreParameters	0.000010
javax.crypto.MacSpi	0.000010
java.security.spec.DSAPrivateKeySpec	0.000010
java.security.KeyPairGeneratorSpi	0.000010
java.security.UnresolvedPermission	0.000010
javax.crypto.KeyGeneratorSpi	0.000010
javax.net.ssl.SSLContextSpi	0.000010
java.security.KeyStoreSpi	0.000010
java.security.cert.CertPathBuilderSpi	0.000010
java.beans.PropertyChangeListener	0.000010
java.io.StreamTokenizer	0.000010
java.security.AlgorithmParameterGeneratorSpi	0.000010
java.security.spec.RSAPrivateCrtKeySpec	0.000010
javax.crypto.spec.PBEParameterSpec	0.000010
java.lang.StrictMath	0.000010
java.util.prefs.AbstractPreferences	0.000010
java.nio.channels.NonWritableChannelException	0.000010
javax.net.ssl.CertPathTrustManagerParameters	0.000010
java.security.cert.CertStore	0.000010
java.util.EventListenerProxy	0.000010
java.util.logging.ErrorManager	0.000010
java.security.cert.CertPathValidator	0.000010
java.net.CacheResponse	0.000010
java.security.acl.Owner	0.000010
javax.crypto.KeyAgreement	0.000010
javax.security.cert.CertificateEncodingException	0.000010
java.security.SignatureSpi	0.000010
java.io.PipedWriter	0.000010
java.security.PermissionsHash	0.000010
java.security.cert.CertPathValidatorSpi	0.000010
java.sql.SQLClientInfoException	0.000010
java.nio.BaseByteBuffer	0.000010
java.security.acl.Acl	0.000010
java.security.Timestamp	0.000010

java.io.FilePermissionCollection	0.000010
javax.net.ssl.KeyManagerFactorySpi	0.000010
java.security.cert.PKIXCertPathValidatorResult	0.000010
java.lang.StackOverflowError	0.000010
java.sql.RowIdLifetime	0.000010
java.util.prefs.InvalidPreferencesFormatException	0.000010
java.security.Signer	0.000010
java.security.cert.CertStoreSpi	0.000010
javax.crypto.SecretKeyFactorySpi	0.000010
java.net.HttpRetryException	0.000009
java.net.InterfaceAddress	0.000009
java.util.prefs.FilePreferencesImpl	0.000009
java.util.TooManyListenersException	0.000009
java.security.cert.CertPathBuilder	0.000009
javax.net.ssl.SSLProtocolException	0.000009
java.lang.UnknownError	0.000009
java.util.jar.JarVerifier	0.000009
javax.crypto.spec.DHPublicKeySpec	0.000009
java.util.PriorityQueue	0.000009
java.net.SocketPermissionCollection	0.000009
java.text.CollationElementIterator	0.000009
java.util.prefs.XMLParser	0.000009
javax.security.auth.login.LoginException	0.000009
javax.security.cert.Certificate	0.000009
java.security.KeyFactorySpi	0.000009
java.security.cert.PKIXBuilderParameters	0.000009
java.security.interfaces.RSAMultiPrimePrivateCrtKey	0.000009
javax.net.DefaultServerSocketFactory	0.000009
javax.crypto.spec.OAEPParameterSpec	0.000009
java.util.logging.SimpleFormatter	0.000009
java.net.JarURLConnection	0.000009
java.security.interfaces.RSAPrivateCrtKey	0.000009
java.text.RuleBasedBreakIterator	0.000009
java.security.Certificate	0.000009
java.sql.ResultSetConcurrency	0.000009
java.io.LineNumberInputStream	0.000009
java.sql.ParameterMetaData	0.000009
java.util.MiniEnumSet	0.000009
javax.security.auth.callback.PasswordCallback	0.000009
javax.crypto.spec.DHPrivateKeySpec	0.000009
javax.crypto.CipherInputStream	0.000009
javax.security.auth.PrivateCredentialPermission	0.000009
java.util.HugeEnumSet	0.000009

javax.crypto.interfaces.DHPublicKey	0.000009
javax.net.DefaultSocketFactory	0.000009
java.security.cert.LDAPCertStoreParameters	0.000009
java.security.spec.RSAKeyGenParameterSpec	0.000009
javax.crypto.spec.DESedeKeySpec	0.000009
javax.sql.RowSetListener	0.000009
java.util.jar.InitManifest	0.000009
javax.crypto.KeyAgreementSpi	0.000009
javax.crypto.interfaces.DHPrivateKey	0.000009
java.util.jar.JarInputStream	0.000009
java.sql.ClientInfoStatus	0.000009
java.security.acl.AclNotFoundException	0.000009
javax.crypto.CipherOutputStream	0.000009
java.nio.channels.NonReadableChannelException	0.000009
java.lang.UnsupportedClassVersionError	0.000009
java.lang.ClassCircularityError	0.000009
javax.net.ssl.X509ExtendedKeyManager	0.000009
java.security.SecureClassLoader	0.000009
javax.security.auth.Destroyable	0.000009
java.text.ChoiceFormat	0.000009
java.net.ContentHandlerFactory	0.000009
java.sql.SQLData	0.000009
java.nio.channels.OverlappingFileLockException	0.000009
java.awt.font.TextAttribute	0.000009
java.text.Bidi	0.000009
java.net.DatagramSocketImplFactory	0.000009
javax.security.cert.CertificateExpiredException	0.000009
javax.security.cert.CertificateNotYetValidException	0.000009
java.nio.channels.NoConnectionPendingException	0.000009
java.nio.channels.NotYetConnectedException	0.000009
javax.net.ssl.SSLSessionBindingListener	0.000009
javax.crypto.interfaces.PBEKey	0.000009
java.nio.channels.Channels	0.000009
javax.crypto.ExemptionMechanismSpi	0.000009
javax.crypto.ExemptionMechanism	0.000009
java.net.ProxySelector	0.000009
java.sql.SQLPermission	0.000009
java.security.interfaces.DSAKeyPairGenerator	0.000009
java.util.SpecialAccess	0.000009
java.net.ProtocolException	0.000009
java.nio.channels.NotYetBoundException	0.000008
java.security.cert.PKIXCertPathChecker	0.000008
java.io.SequenceInputStream	0.000008

java.lang.Compiler	0.000008
java.util.prefs.FilePreferencesFactoryImpl	0.000008
java.text.StringCharacterIterator	0.000008
java.security.DigestInputStream	0.000008
java.nio.DirectByteBuffers	0.000008
java.security.ProviderException	0.000008
java.util.zip.InflaterOutputStream	0.000008
java.nio.channels.UnresolvedAddressException	0.000008
java.nio.channels.UnsupportedAddressTypeException	0.000008
java.net.CookieHandler	0.000008
java.util.logging.FileHandler	0.000008
java.nio.LongToByteBufferAdapter	0.000008
java.nio.DoubleToByteBufferAdapter	0.000008
java.nio.CharToByteBufferAdapter	0.000008
java.lang.annotation.AnnotationFormatError	0.000008
java.security.spec.PSSParameterSpec	0.000008
java.lang.EnumConstantNotPresentException	0.000008
java.lang.reflect.UndeclaredThrowableException	0.000008
java.lang.annotation.IncompleteAnnotationException	0.000008
javax.crypto.spec.DHGenParameterSpec	0.000008
java.nio.FloatToByteBufferAdapter	0.000008
java.nio.IntToByteBufferAdapter	0.000008
java.nio.ShortToByteBufferAdapter	0.000008
java.net.ProxySelectorImpl	0.000008
java.lang.annotation.AnnotationTypeMismatchException	0.000008
java.net.UnknownServiceException	0.000008
java.security.cert.X509CRLSelector	0.000008
java.nio.channels.ClosedSelectorException	0.000008
java.net.CookieStore	0.000008
java.util.zip.DeflaterInputStream	0.000008
java.sql.SQLFeatureNotSupportedException	0.000008
java.nio.channels.AlreadyConnectedException	0.000008
java.util.zip.CheckedInputStream	0.000008
java.awt.font.NumericShaper	0.000008
javax.net.ssl.SSLPermission	0.000008
javax.net.ssl.DefaultSSLSocketFactory	0.000008
java.net.DefaultFileNameMap	0.000008
javax.net.ssl.DefaultHostnameVerifier	0.000008
javax.net.ssl.DefaultSSLServerSocketFactory	0.000008
java.util.logging.XMLFormatter	0.000008
java.util.InputMismatchException	0.000008
java.net.ResponseCache	0.000008
javax.net.ssl.SSLKeyException	0.000008

javax.crypto.spec.RC2ParameterSpec	0.000008
java.security.cert.PolicyQualifierInfo	0.000008
java.net.CookieStoreImpl	0.000008
java.security.UnresolvedPermissionCollection	0.000008
java.io.StringBufferInputStream	0.000008
java.security.cert.PKIXCertPathBuilderResult	0.000008
java.nio.channels.ConnectionPendingException	0.000008
java.util.jar.JarException	0.000008
java.lang.UnsafeByteSequence	0.000008
java.beans.PropertyChangeSupport	0.000008
java.security.DigestOutputStream	0.000008
java.security.KeyRep	0.000008
java.nio.MappedByteBufferAdapter	0.000008
java.util.logging.MemoryHandler	0.000008
java.util.UnknownFormatFlagsException	0.000008
java.io.CharConversionException	0.000008
javax.sql.RowSetWriter	0.000008
javax.sql.ConnectionPoolDataSource	0.000008
javax.sql.RowSetReader	0.000008
javax.security.cert.CertificateParsingException	0.000008
javax.security.auth.SubjectDomainCombiner	0.000008
java.util.ArrayDeque	0.000008
java.security.spec.ECGenParameterSpec	0.000008
java.nio.channels.FileLockInterruptionException	0.000008
java.net.MulticastSocket	0.000008
javax.sql.DataSource	0.000008
javax.net.ssl.KeyStoreBuilderParameters	0.000008
java.util.zip.CheckedOutputStream	0.000008
java.net.CookieManager	0.000008
java.net.CookiePolicy	0.000008
javax.crypto.spec.RC5ParameterSpec	0.000008
javax.crypto.EncryptedPrivateKeyInfo	0.000008
java.util.logging.LoggingPermission	0.000008
java.util.logging.LoggingMXBean	0.000008
java.util.logging.SocketHandler	0.000008
java.sql.DataTruncation	0.000008
java.security.spec.ECPrivateKeySpec	0.000008
java.security.spec.ECPublicKeySpec	0.000008
java.security.spec.RSAMultiPrimePrivateCrtKeySpec	0.000008
java.security.AuthProvider	0.000008
java.security.GuardedObject	0.000008
java.util.jar.Pack200	0.000008
java.net.MulticastGroupRequest	0.000008

java.beans.PropertyChangeListenerProxy	0.000007
java.beans.IndexedPropertyChangeEvent	0.000007
java.security.SignedObject	0.000007
java.net.SocketUtils	0.000007
javax.crypto.SealedObject	0.000007
java.util.prefs.NodeSet	0.000007
java.net.PortUnreachableException	0.000007
java.sql.SQLDataException	0.000007
java.sql.SQLInvalidAuthorizationSpecException	0.000007
java.nio.NIOAccess	0.000007
java.sql.SQLIntegrityConstraintViolationException	0.000007
java.lang.LangAccessImpl	0.000007
java.sql.ResultSetHoldability	0.000007
java.net.SecureCacheResponse	0.000007
javax.net.Muffins	0.000007
java.sql.SQLNonTransientConnectionException	0.000007
java.util.zip.ZipError	0.000007
java.sql.SQLTransientConnectionException	0.000007
java.sql.ResultSetType	0.000007
java.security.interfaces.ECPublicKey	0.000007
java.sql.SQLTransactionRollbackException	0.000007
java.sql.SQLRecoverableException	0.000007
java.security.interfaces.ECPrivateKey	0.000007
java.net.NoRouteToHostException	0.000007
java.sql.SQLTimeoutException	0.000007
java.sql.SQLSyntaxErrorException	0.000007
java.util.logging.ConsoleHandler	0.000007

APPENDIX E – PHP Script for Parsing HTML Documentation of Java SE 5 API Packages

```
<pre>

<?php

/*
 * The purpose of this script is to read java doc detailing the J2SE API
packages, classes and methods
 *  and parse it to get package name, class name, and method information
 *  and generate a csv with this info for each JAVA api level from 1.0
to 1.8.
 *
 * Here is data dictionary of CSV file:
 *
 * package_name
 * package_section
 * class_name
 * class_full
 * class_section
 * method_name
 * method_full
 * method_para
 * version
 *
 */

/*
 * This script uses library called simpledomparser.php, which is a
simple PHP HTML DOM parser written in PHP5+, supports invalid HTML, and
provides a very easy way to handle HTML elements.
 */

require('simpledomparser.php');

/*
 * This method calculate similarity between given text strings.
 */

function text_similarity($string_a, $string_b){

    $similar_text = '';

    similar_text($string_a, $string_b, $percent);

    $similar_text_a = (100.0 - $percent) * 1.0 *
strlen($string_a.$string_b);

    similar_text($string_b, $string_a, $percent);

    $similar_text_b = (100.0 - $percent) * 1.0 *
strlen($string_a.$string_b);

    return ($similar_text_a + $similar_text_b) / 200.0;

}
```

```
 50
 51 /*
 52  * This method strips/removes and replace special HTML charactors like
end of line, tab, space from given string.
 53  */
 54
 55 function clean_string($result_value){
 56
 57     $result_value = preg_replace('~[\r\n]+~', '', $result_value);
 58     $result_value = preg_replace('~[\t]+~', '', $result_value);
 59
 60     if(strlen($result_value) > 0){
 61
 62         $result_value = str_replace(' ', ' ', $result_value);
 63
 64         if(strlen($result_value) > 0){
 65
 66             $result_value = str_replace('nbsp;', ' ', $result_value);
 67
 68         }
 69
 70     }
 71
 72     while(substr_count($result_value, '  ') > 0){
 73
 74         $result_value = str_replace('  ', ' ', $result_value);
 75
 76     }
 77
 78     $result_value = str_replace('&gt;', '>', $result_value);
 79     $result_value = str_replace('&lt;', '<', $result_value);
 80
 81     return $result_value;
 82
 83
 84 }
 85
 86 /*
 87  * This method gets substring between 2 given index or string from given
text
 88  *
 89  */
 90
 91 function data_search($text,$start,$end){
 92
 93 $start_len = strlen($start);
 94
 95 $pos1 = strpos($text, $start) + $start_len;
 96
 97 $cut_page = substr($text,$pos1);
 98
 99 $pos2 = strpos($cut_page, $end);
100
101 $data = substr($cut_page,0,$pos2);
102
103 return $data;
104
```

```
105 }
106
107 /*
108  * This method creates list of package names
109  * and their respective url location which contains list of classes and
their discription.
110 */
111
112 function file_scrape($url){
113
114     /*
115      * this section reads html page from given link and creates DOM
objects with nodes and their attributes
116      */
117
118     $html_body = file_get_contents($url);
119     $dom = new simple_html_dom(null);
120     $dom->load($html_body);
121     $html = $dom;
122
123     $result_table = array();
124
125
126
127     if(true){
128
129
130 #        $package_url_table = ;
131
132         foreach($html->find('body table.overviewSummary')[0]->find('a')
as $url){
133
134             if(strpos($url, 'package-summary') !== false){
135
136                 $url_arr[] = array($url->href, $url->plaintext);
137
138             }
139
140         }
141         /*
142         $package_url_tbody = $package_url_table->find('tbody')[1];
143
144         echo $package_url_table;
145
146         foreach($package_url_tbody as $package_url_tr){
147
148             $package_url_td = $package_url_tr->find('td')[0];
149             $package_url = $package_url_td->find('a')[0]->href;
150
151             echo $package_url;
152
153         }
154         */
155
156     }
157
158     return $url_arr;
```

```
159
160 }
161
162
163 /*
164  * This method parses package page from java doc using html dom parser
and extract all information like,
165  * summary of classes and interface and also description of classes and
interface.
166  * It calls internal method for each class found in given package and
run class analysis for that class as well as all nested classes within that
class.
167  *
168  */
169
170
171 function scrape($url, $name, $fp, $test, $url_stub){
172
173     /*
174      * this section reads html page from given link and creates DOM
objects with nodes and their attributes
175      */
176
177     $url = $url_stub.$url;
178
179     $html_body = file_get_contents($url);
180     $dom = new simple_html_dom(null);
181     $dom->load($html_body);
182     $html = $dom;
183
184     $result_table = array();
185
186
187
188     if(true){
189
190
191         $test_limit = 1;
192         $test_count = 0;
193
194         foreach($html->find('body div.contentContainer ul.blockList li')
as $li){
195
196             $this_table_title = "";
197
198             if(strlen($li->find('table caption span')[0]->plaintext) >
0){
199
200                 $this_table_title = str_replace(' Summary', '', $li-
>find('table caption span')[0]->plaintext);
201
202             }
203
204             if(strlen($this_table_title) > 0){
205
206                 $this_table_title = str_replace(' ', '',
$this_table_title);
```

```
 207
 208             }
 209
 210
 211
 212             foreach($li->find('table tbody tr td.colFirst a') as
$url_html){
 213
 214                 $class_url = $url_html->href;
 215                 $class_name = $url_html->plaintext;
 216                 $class_url = str_replace('../../', '', $url_html->href);
 217                 $class_url = str_replace('../', '', $class_url);
 218                 $class_url = str_replace('../', '', $class_url);
 219                 $class_url = str_replace('../', '', $class_url);
 220                 $class_url = str_replace('../', '', $class_url);
 221                 $class_url = str_replace('../', '', $class_url);
 222                 $class_url = str_replace('../', '', $class_url);
 223                 $class_url = str_replace('../', '', $class_url);
 224
 225                 $java_substr = 'java/';
 226                 $javax_substr = 'javax/';
 227                 $org_substr = 'org/';
 228
 229                 $java_pos = 0;
 230                 $javax_pos = 0;
 231                 $org_pos = 0;
 232
 233                 if(strpos($class_url, $java_substr) !== false){
 234
 235                     $java_pos = strpos($class_url, $java_substr);
 236
 237                 }
 238
 239                 if(strpos($class_url, $javax_substr) !== false){
 240
 241                     $javax_pos = strpos($class_url, $javax_substr);
 242
 243                 }
 244
 245                 if(strpos($class_url, $org_substr) !== false){
 246
 247                     $org_pos = strpos($class_url, $org_substr);
 248
 249                 }
 250
 251                 $class_pos = 0;
 252
 253                 $class_pos_arr = array();
 254
 255                 $class_pos_arr[] = $java_pos;
 256                 $class_pos_arr[] = $javax_pos;
 257                 $class_pos_arr[] = $org_pos;
 258
 259                 #var_dump($class_pos_arr);
 260
 261                 #echo sizeof($class_pos_arr);
 262
```

```
263                 $class_pos_value_arr = array();
264
265                 foreach($class_pos_arr as $this_class_pos){
266
267                     if($this_class_pos > 0){
268
269                         $class_pos_value_arr[] = $this_class_pos;
270
271                     }
272
273                 }
274
275                 $class_pos_min = 0;
276
277                 if(sizeof($class_pos_value_arr) > 1){
278
279                     $class_pos_min = min($class_pos_value_arr);
280
281                     #var_dump(sizeof($class_pos_value_arr))."bajs";
282
283                 }else{
284
285                     if(sizeof($class_pos_value_arr) > 0){
286
287                         $class_pos_min =
intval($class_pos_value_arr[0]);
288
289
290
291                     }
292                 }
293
294                 if($class_pos_min > 0){
295
296                     $class_pos = $class_pos_min;
297
298                 }
299
300
301
302                 $class_url = substr($class_url, $class_pos,
strlen($class_url));
303
304                 $class_url = $url_stub.$class_url;
305
306                 $temp_arr = array($name, $url, $this_table_title,
$class_name, $class_url);
307
308                 #var_dump($temp_arr);
309
310                 #var_dump($temp_arr);
311
312                 $nested_class_arr = class_scrape($class_url,
$class_name, $temp_arr, $fp, $test, $url_stub);
313
314                 foreach($nested_class_arr as $nested_class){
315
```

```
 316                     $nested_class_url = $nested_class[0];
 317                     $nested_class_name = $nested_class[1];
 318
 319                     $nested_temp_arr = array($name, $url,
$this_table_title, $nested_class_name, $nested_class_url);
 320
 321                     $nested_class_arr_2 =
class_scrape($nested_class_url, $nested_class_name, $nested_temp_arr, $fp,
$test, $url_stub);
 322
 323                     if(sizeof($nested_class_arr_2) > 0){
 324
 325                         #echo 'nested_nested';
 326
 327                         foreach($nested_class_arr_2 as $nested_class_2){
 328
 329                             $nested_class_url_2 = $nested_class_2[0];
 330                             $nested_class_name_2 = $nested_class_2[1];
 331
 332                             $nested_temp_arr_2 = array($name, $url,
$this_table_title, $nested_class_name_2, $nested_class_url_2);
 333
 334                             $nested_class_arr_3 =
class_scrape($nested_class_url_2, $nested_class_name_2, $nested_temp_arr_2,
$fp, $test, $url_stub);
 335
 336                             if(sizeof($nested_class_arr_3) > 0){
 337
 338                                 echo 'nested_nested_nested';
 339
 340                                 # there are 4 cases of this
 341
 342                             }
 343
 344
 345                         }
 346
 347                     }
 348
 349                 }
 350
 351             }
 352
 353         }
 354
 355
 356     }
 357
 358
 359 }
 360
 361
 362 /*
 363  * This method parses class page from java doc using html dom parser and
extract all information like,
 364  * nested classe, field summary, Constructor details, and methods list
from class using dom nodes and attribute
```

```
 365  * and writes into given CSV file. it also prepares list of nested
classes to be processed and return the list.
 366  */
 367
 368
 369 function class_scrape($url, $name, $temp_arr, $fp, $test, $url_stub){
 370
 371     $class_section = "";
 372     $method_name = "";
 373     $method_full = "";
 374     $method_para = "";
 375
 376     $package_name = $temp_arr[0];
 377     $version = $url_stub;
 378     $package_section = $temp_arr[2];
 379     $class_name = $temp_arr[3];
 380
 381     $count_add = 0;
 382
 383     $nested_url_arr = array();
 384
 385     $result_table = array();
 386     $data_table = array();
 387     $class_table = array();
 388
 389     $print_table = array();
 390
 391     $inside_table = false;
 392
 393     if(strpos(strtolower($url), strtolower('applet/Applet.html')) !==
false && $test || !$test){
 394
 395         $html_body = file_get_contents($url);
 396
 397
 398
 399         #$table_start_remove = '<TABLE BORDER="0" CELLPADDING="0"
CELLSPACING="0" SUMMARY="">'."\n".'<TR ALIGN="right" VALIGN="">'."\n".'<TD
NOWRAP><FONT SIZE="-1">'."\n".'<CODE>';
 400         #$table_end_remove =
'</TD>'."\n".'</TR>'."\n".'</TABLE>'."\n".'</CODE>';
 401
 402         #$html_body = str_replace($table_start_remove,'',$html_body);
 403         #$html_body =
str_replace($table_end_remove,'</CODE>',$html_body);
 404
 405
 406
 407         $dom = new simple_html_dom(null);
 408         $dom->load($html_body);
 409         $html = $dom;
 410
 411         #echo $html;
 412
 413
 414         if(true){
 415
```

```
 416
 417
 418
 419             $class_full = "";
 420
 421             $class_full = $html->find('div.contentContainer
div.description ul.blockList li.blockList pre')[0]->plaintext;
 422
 423
 424
 425             foreach($html->find('div.contentContainer div.details
ul.blockList li.blockList ul.blockList h3') as $section_detail){
 426
 427                 $class_section = str_replace(' Detail', '',
$section_detail->plaintext);
 428
 429                 foreach($section_detail->parent()->find('ul.blockList
li.blockList') as $method_detail){
 430
 431                     #echo $method_detail;
 432
 433                     #$method_dom = new simple_html_dom(null);
 434                     #$method_dom->load($method_detail);
 435
 436                     #echo $method_dom;
 437
 438                     $method_full = $method_detail->find('pre')[0]-
>plaintext;
 439                     $method_name = $method_detail->find('h4')[0]-
>plaintext;
 440
 441
 442
 443                     $method_para = "";
 444
 445                     if(strtolower($class_section) == 'field'){
 446
 447                         $package_name_url = str_replace('.', '/',
$package_name);
 448
 449                         $class_name_url = $class_name;
 450
 451                         if(strpos($class_name, '.') !== false){
 452
 453                             $class_name_url = explode('.',
$class_name)[0];
 454
 455                         }
 456
 457                         $url_stub_url = str_replace('C:/doc', 'C:/jdk-
source-doc', $url_stub);
 458
 459                         $url_stub_url = str_replace('/doc/', '/',
$url_stub_url);
 460
 461                         $source_file_url =
$url_stub_url.$package_name_url.'/'.$class_name_url.'.java';
```

```
 462
 463                          $source_file = file($source_file_url);
 464
 465                          $class_source_lines = array();
 466                          $method_source_lines = array();
 467
 468                          $class_found = false;
 469
 470                          foreach($source_file as $source_row){
 471
 472                              if(strpos($source_row, $class_name_url) !==
false){
 473
 474                                  $class_found = true;
 475
 476                              }
 477
 478                              if($class_found == true){
 479
 480                                  #$class_source_lines[] = $source_row;
 481
 482                                  if(strpos($source_row, ' '.$method_name)
!== false){
 483
 484                                      $method_source_lines[] =
$source_row;
 485
 486                                  }else{
 487
 488                                      #echo 'could not find method line';
 489
 490                                      #var_dump($source_row);
 491                                      #var_dump($method_name);
 492
 493                                  }
 494
 495                              }
 496
 497                          }
 498
 499                          if(sizeof($method_source_lines) == 0){
 500
 501                              #echo 'could not find method';
 502
 503
#var_dump($source_file_url.':'.$package_name.'@'.$class_name.'#'.$method_name
);
 504
 505                          }
 506
 507                          $method_declaration_line = "";
 508
 509                          if(sizeof($method_source_lines) == 1){
 510
 511                              $method_declaration_line =
$method_source_lines[0];
 512
```

```
513                         }else{
514
515                             #echo "bajs";
516
517                             $min_key = 9999;
518                             $min_key_2 = 9999;
519
520                             $method_equals_source_lines = array();
521
522                             $key_arr = array();
523                             $key_arr_2 = array();
524
525                             foreach($method_source_lines as
$this_method_source_line){
526
527                                 if(strpos($this_method_source_line, ";")
!== false){
528
529
530                                     $key = text_similarity($method_name,
$this_method_source_line);
531                                     if(!isset($key_arr[$key])){
532
533                                         $key_arr[$key] =
534                                         $this_method_source_line;
535                                         if($key < $min_key){
536
537                                             $min_key = $key;
538                                         }
539                                     }else{
540                                         #echo 'two declaration lines
541                                         have the same similarity';
542                                     }
543                                     if(sizeof($key_arr) > 0){
544                                         $method_declaration_line =
545                                         $key_arr[$min_key];
546                                         if(!isset($key_arr[$min_key])){
547
548                                             #var_dump($key_arr);
549                                         }
550                                     }
551                                     if(strpos($this_method_source_line,
552                                     "=") !== false){
553
554
555
556
557
558
559
560
561
```

```
562                                         $method_equals_source_lines[] =
$this_method_source_line;
563
564                                     }
565
566
if(sizeof($method_equals_source_lines) == 1){
567
568                                         $method_declaration_line =
$method_equals_source_lines[0];
569
570                                     }else{
571
572
foreach($method_equals_source_lines as $this_method_equals_line){
573
574                                             $key_2 =
text_similarity($method_name, $this_method_equals_line);
575
576
577
if(!isset($key_arr_2[$key_2])){
578
579                                                 $key_arr_2[$key_2] =
$this_method_equals_line;
580
581                                                 if($key_2 < $min_key_2){
582
583                                                     $min_key_2 = $key_2;
584
585                                                 }
586
587                                             }else{
588
589                                                 #echo 'two declaration
lines have the same similarity at equals level';
590
591                                             }
592
593                                             if(sizeof($key_arr_2) > 0){
594
595                                                 $method_declaration_line
= $key_arr_2[$min_key_2];
596
597                                             }
598
599
600                                         }
601
602                                     }
603
604
605                                 }
606
607
608
609
```

```
 610
 611                                }
 612
 613
 614                                foreach($method_source_lines as
$this_method_source_line_2){
 615
 616                                    $method_full_last_call = str_replace('
', '', clean_string($method_full));
 617                                    $this_method_last_call = str_replace('
', '', clean_string($this_method_source_line_2));
 618
 619                                    if(strpos($this_method_last_call,
$method_full_last_call) !== false){
 620
 621                                        $method_declaration_line =
$this_method_source_line_2;
 622
 623                                        #echo "good";
 624
 625                                    }
 626
 627                                }
 628
 629
 630
 631                            }
 632
 633                            #echo $method_declaration_line;
 634
 635                            if(strpos($method_declaration_line, '=') !==
false){
 636
 637                                $eq_pos = strpos($method_declaration_line,
'=');
 638
 639                                $decl_eq = substr($method_declaration_line,
$eq_pos, strlen($method_declaration_line));
 640
 641                                $semi_pos = strpos($decl_eq, ';');
 642
 643                                $decl_semi = substr($decl_eq, 0, $semi_pos);
 644
 645                                $para = "";
 646
 647                                if(strpos($decl_semi, '= ') !== false){
 648
 649                                    $para = str_replace('= ', '',
$decl_semi);
 650
 651                                }else{
 652
 653                                    $para = str_replace('=', '',
$decl_semi);
 654
 655                                }
 656
```

```
657                             #var_dump($para);
658
659
660                             $method_para = $para;
661
662
663                         }
664
665
666                     }
667
668
669                     $result_row = array($package_name, $package_section,
$class_name, $class_full, $class_section, $method_name, $method_full,
$method_para, $version);
670
671                     $print_row = array();
672
673                     foreach($result_row as $result_value){
674
675                         #$result_value = preg_replace ('%[\x00-
\x30\x127]%', '', $result_value);
676                         $result_value = preg_replace('~[\r\n]+~', '',
$result_value);
677                         $result_value = preg_replace('~[\t]+~', '',
$result_value);
678
679                         if(strlen($result_value) > 0){
680
681                             $result_value = str_replace(' ', ' ',
$result_value);
682
683                             if(strlen($result_value) > 0){
684
685                                 $result_value = str_replace('nbsp;', '
', $result_value);
686
687                             }
688
689                         }
690
691                         while(substr_count($result_value, '  ') > 0){
692
693                             $result_value = str_replace('  ', ' ',
$result_value);
694
695                         }
696
697                         $result_value = str_replace('&gt;', '>',
$result_value);
698                         $result_value = str_replace('&lt;', '<',
$result_value);
699
700
701
702                         $print_row[] = $result_value;
703
```

```
 704                     }
 705
 706                     $print_table[] = $print_row;
 707
 708
 709
 710                     $count_add++;
 711
 712                 }
 713
 714                 foreach($section_detail->parent()-
>find('ul.blockListLast li.blockList') as $method_detail){
 715
 716                     #echo $method_detail;
 717
 718                     #$method_dom = new simple_html_dom(null);
 719                     #$method_dom->load($method_detail);
 720
 721                     #echo $method_dom;
 722
 723                     $method_full = $method_detail->find('pre')[0]-
>plaintext;
 724                     $method_name = $method_detail->find('h4')[0]-
>plaintext;
 725
 726
 727
 728                     $method_para = "";
 729
 730                     if(strtolower($class_section) == 'field'){
 731
 732                         $package_name_url = str_replace('.', '/',
$package_name);
 733
 734                         $class_name_url = $class_name;
 735
 736                         if(strpos($class_name, '.') !== false){
 737
 738                             $class_name_url = explode('.',
$class_name)[0];
 739
 740                         }
 741
 742                         $url_stub_url = str_replace('C:/doc', 'C:/jdk-
source-doc', $url_stub);
 743
 744                         $url_stub_url = str_replace('/doc/', '/',
$url_stub_url);
 745
 746                         $source_file_url =
$url_stub_url.$package_name_url.'/'.$class_name_url.'.java';
 747
 748                         $source_file = file($source_file_url);
 749
 750                         $class_source_lines = array();
 751                         $method_source_lines = array();
 752
```

```
 753                             $class_found = false;
 754
 755                             foreach($source_file as $source_row){
 756
 757                                 if(strpos($source_row, $class_name_url) !==
false){
 758
 759                                     $class_found = true;
 760
 761                                 }
 762
 763                                 if($class_found == true){
 764
 765                                     #$class_source_lines[] = $source_row;
 766
 767                                     if(strpos($source_row, ' '.$method_name)
!== false){
 768
 769                                         $method_source_lines[] =
$source_row;
 770
 771                                     }else{
 772
 773                                         #echo 'could not find method line';
 774
 775                                         #var_dump($source_row);
 776                                         #var_dump($method_name);
 777
 778                                     }
 779
 780                                 }
 781
 782                             }
 783
 784                             if(sizeof($method_source_lines) == 0){
 785
 786                                 #echo 'could not find method';
 787
 788
#var_dump($source_file_url.':'.$package_name.'@'.$class_name.'#'.$method_name
);
 789
 790                             }
 791
 792                             $method_declaration_line = "";
 793
 794                             if(sizeof($method_source_lines) == 1){
 795
 796                                 $method_declaration_line =
$method_source_lines[0];
 797
 798                             }else{
 799
 800                                 #echo "bajs";
 801
 802                                 $min_key = 9999;
 803                                 $min_key_2 = 9999;
```

```
 804
 805                                $method_equals_source_lines = array();
 806
 807                                $key_arr = array();
 808                                $key_arr_2 = array();
 809
 810                                foreach($method_source_lines as
$this_method_source_line){
 811
 812                                    if(strpos($this_method_source_line, ";")
!== false){
 813
 814
 815                                        $key = text_similarity($method_name,
$this_method_source_line);
 816
 817                                        if(!isset($key_arr[$key])){
 818
 819                                            $key_arr[$key] =
$this_method_source_line;
 820
 821                                            if($key < $min_key){
 822
 823                                                $min_key = $key;
 824
 825                                            }
 826
 827                                        }else{
 828
 829                                            #echo 'two declaration lines
have the same similarity';
 830
 831                                        }
 832
 833                                        if(sizeof($key_arr) > 0){
 834
 835                                            $method_declaration_line =
$key_arr[$min_key];
 836
 837                                            if(!isset($key_arr[$min_key])){
 838
 839                                                #var_dump($key_arr);
 840
 841                                            }
 842
 843                                        }
 844
 845                                        if(strpos($this_method_source_line,
"=") !== false){
 846
 847                                            $method_equals_source_lines[] =
$this_method_source_line;
 848
 849                                        }
 850
 851
if(sizeof($method_equals_source_lines) == 1){
```

```
 852
 853                                          $method_declaration_line =
$method_equals_source_lines[0];
 854
 855                                      }else{
 856
 857
foreach($method_equals_source_lines as $this_method_equals_line){
 858
 859                                                  $key_2 =
text_similarity($method_name, $this_method_equals_line);
 860
 861
 862
if(!isset($key_arr_2[$key_2])){
 863
 864                                                      $key_arr_2[$key_2] =
$this_method_equals_line;
 865
 866                                                      if($key_2 < $min_key_2){
 867
 868                                                          $min_key_2 = $key_2;
 869
 870                                                      }
 871
 872                                                  }else{
 873
 874                                                      #echo 'two declaration
lines have the same similarity at equals level';
 875
 876                                                  }
 877
 878                                                  if(sizeof($key_arr_2) > 0){
 879
 880                                                      $method_declaration_line
= $key_arr_2[$min_key_2];
 881
 882                                                  }
 883
 884
 885                                          }
 886
 887                                      }
 888
 889
 890                                  }
 891
 892
 893
 894
 895
 896                              }
 897
 898
 899                              foreach($method_source_lines as
$this_method_source_line_2){
 900
```

```
 901                                     $method_full_last_call = str_replace('
', '', clean_string($method_full));
 902                                     $this_method_last_call = str_replace('
', '', clean_string($this_method_source_line_2));
 903
 904                                     if(strpos($this_method_last_call,
$method_full_last_call) !== false){
 905
 906                                         $method_declaration_line =
$this_method_source_line_2;
 907
 908                                         #echo "good";
 909
 910                                     }
 911
 912                                 }
 913
 914
 915
 916                             }
 917
 918                             #echo $method_declaration_line;
 919
 920                             if(strpos($method_declaration_line, '=') !==
false){
 921
 922                                 $eq_pos = strpos($method_declaration_line,
'=');
 923
 924                                 $decl_eq = substr($method_declaration_line,
$eq_pos, strlen($method_declaration_line));
 925
 926                                 $semi_pos = strpos($decl_eq, ';');
 927
 928                                 $decl_semi = substr($decl_eq, 0, $semi_pos);
 929
 930                                 $para = "";
 931
 932                                 if(strpos($decl_semi, '= ') !== false){
 933
 934                                     $para = str_replace('= ', '',
$decl_semi);
 935
 936                                 }else{
 937
 938                                     $para = str_replace('=', '',
$decl_semi);
 939
 940                                 }
 941
 942                                 #var_dump($para);
 943
 944
 945                                 $method_para = $para;
 946
 947
 948                             }
```

```
 949
 950
 951                     }
 952
 953
 954                     $result_row = array($package_name, $package_section,
$class_name, $class_full, $class_section, $method_name, $method_full,
$method_para, $version);
 955
 956                     $print_row = array();
 957
 958                     foreach($result_row as $result_value){
 959
 960                         #$result_value = preg_replace ('%[\x00-
\x30\x127]%', '', $result_value);
 961                         $result_value = preg_replace('~[\r\n]+~', '',
$result_value);
 962                         $result_value = preg_replace('~[\t]+~', '',
$result_value);
 963
 964                         if(strlen($result_value) > 0){
 965
 966                             $result_value = str_replace(' ', ' ',
$result_value);
 967
 968                             if(strlen($result_value) > 0){
 969
 970                                 $result_value = str_replace('nbsp;', '
', $result_value);
 971
 972                             }
 973
 974                         }
 975
 976                         while(substr_count($result_value, '  ') > 0){
 977
 978                             $result_value = str_replace('  ', ' ',
$result_value);
 979
 980                         }
 981
 982                         $result_value = str_replace('&gt;', '>',
$result_value);
 983                         $result_value = str_replace('&lt;', '<',
$result_value);
 984
 985
 986
 987                         $print_row[] = $result_value;
 988
 989                     }
 990
 991                     $print_table[] = $print_row;
 992
 993
 994
 995                     $count_add++;
```

```
 996
 997                 }
 998
 999
1000
1001             }
1002
1003
1004
1005             #echo $html;
1006
1007
1008
1009             foreach($html->find('div.contentContainer div.summary
ul.blockList li.blockList ul.blockList li.blockList h3') as
$summary_section){
1010
1011                 #echo $summary_section;
1012
1013                 if(strpos(strtolower($summary_section->plaintext),
'summary') !== false){
1014
1015                     if(strpos(strtolower($summary_section->plaintext),
'nested') !== false){
1016
1017                         #foreach($summary_section->parent()->find('table
tbody tr td.colLast code span a') as $nested_link){
1018
1019                         foreach($summary_section->parent()->find('table
tbody tr td.colLast code') as $nested_link){
1020
1021                             $dom_nested = new simple_html_dom(null);
1022                             $dom_nested->load($nested_link->innertext);
1023
1024                             foreach($dom_nested->find('a') as
$nested_href){
1025
1026                                 $nested_name = $nested_href->plaintext;
1027
1028                                 $nested_url = str_replace('../../', '',
$nested_href->href);
1029                                 $nested_url = str_replace('../', '',
$nested_url);
1030                                 $nested_url = str_replace('../', '',
$nested_url);
1031                                 $nested_url = str_replace('../', '',
$nested_url);
1032                                 $nested_url = str_replace('../', '',
$nested_url);
1033                                 $nested_url = str_replace('../', '',
$nested_url);
1034                                 $nested_url = str_replace('../', '',
$nested_url);
1035                                 $nested_url = str_replace('../', '',
$nested_url);
1036
1037                                 $java_substr = 'java/';
```

```
1038                                    $javax_substr = 'javax/';
1039                                    $org_substr = 'org/';
1040
1041                                    $java_pos = 0;
1042                                    $javax_pos = 0;
1043                                    $org_pos = 0;
1044
1045                                    if(strpos($nested_url, $java_substr) !==
false){
1046
1047                                        $java_pos = strpos($nested_url,
$java_substr);
1048
1049                                    }
1050
1051                                    if(strpos($nested_url, $javax_substr)
!== false){
1052
1053                                        $javax_pos = strpos($nested_url,
$javax_substr);
1054
1055                                    }
1056
1057                                    if(strpos($nested_url, $org_substr) !==
false){
1058
1059                                        $org_pos = strpos($nested_url,
$org_substr);
1060
1061                                    }
1062
1063                                    $nested_pos = 0;
1064
1065                                    $nested_pos_arr = array();
1066
1067                                    $nested_pos_arr[] = $java_pos;
1068                                    $nested_pos_arr[] = $javax_pos;
1069                                    $nested_pos_arr[] = $org_pos;
1070
1071                                    #var_dump($class_pos_arr);
1072
1073                                    #echo sizeof($class_pos_arr);
1074
1075                                    $nested_pos_value_arr = array();
1076
1077                                    foreach($nested_pos_arr as
$this_nested_pos){
1078
1079                                        if($this_nested_pos > 0){
1080
1081                                            $nested_pos_value_arr[] =
$this_nested_pos;
1082
1083                                        }
1084
1085                                    }
1086
```

```
1087                                    $nested_pos_min = 0;
1088
1089                                    if(sizeof($nested_pos_value_arr) > 1){
1090
1091                                        $nested_pos_min =
min($nested_pos_value_arr);
1092
1093
#var_dump(sizeof($class_pos_value_arr))."bajs";
1094
1095                                    }else{
1096
1097                                        if(sizeof($nested_pos_value_arr) >
0){
1098
1099                                            $nested_pos_min =
intval($nested_pos_value_arr[0]);
1100
1101
1102
1103                                        }
1104                                    }
1105
1106                                    if($nested_pos_min > 0){
1107
1108                                        $nested_pos = $nested_pos_min;
1109
1110                                    }
1111
1112
1113
1114                                    $nested_url = substr($nested_url,
$nested_pos, strlen($nested_url));
1115
1116                                    $nested_url = $url_stub.$nested_url;
1117
1118                                    $nested_url_arr[] = array($nested_url,
$nested_name);
1119
1120
1121
1122                                }
1123
1124
1125
1126                            }
1127
1128
1129                        }
1130
1131                    }
1132
1133
1134
1135
1136                }
1137
```

```
1138
1139
1140
1141
1142          }
1143
1144      }
1145
1146
1147      if($count_add == 0){
1148
1149          $result_row = array($package_name, $package_section,
$class_name, $class_full, $class_section, $method_name, $method_full,
$method_para, $version);
1150
1151          $print_row = array();
1152
1153          foreach($result_row as $result_value){
1154
1155              #$result_value = preg_replace ('%[\x00-\x30\x127]%', '',
$result_value);
1156              $result_value = preg_replace('~[\r\n]+~', '',
$result_value);
1157              $result_value = preg_replace('~[\t]+~', '', $result_value);
1158
1159              if(strlen($result_value) > 0){
1160
1161                  $result_value = str_replace(' ', ' ',
$result_value);
1162
1163                  if(strlen($result_value) > 0){
1164
1165                      $result_value = str_replace('nbsp;', ' ',
$result_value);
1166
1167                  }
1168
1169              }
1170
1171              while(substr_count($result_value, '  ') > 0){
1172
1173                  $result_value = str_replace('  ', ' ', $result_value);
1174
1175              }
1176
1177              $result_value = str_replace('&gt;', '>', $result_value);
1178              $result_value = str_replace('&lt;', '<', $result_value);
1179
1180
1181
1182              $print_row[] = $result_value;
1183
1184          }
1185
1186          $print_table[] = $print_row;
1187
1188      }
```

```
1189
1190
1191     foreach($print_table as $this_row){
1192
1193         fputcsv($fp, $this_row);
1194
1195     }
1196
1197
1198     return $nested_url_arr;
1199
1200
1201
1202 }
1203
1204
1205
1206
1207 #error_reporting(0);
1208
1209
1210 /*
1211  * This section creates output csv file with column header.
1212  */
1213
1214 $header_array = array("package_name", "package_url", "package_section",
"class_name", "class_url", "class_section", "jdk_version", "method_full",
"method_type", "method_name", "method", "parameters", "method_version",
"method_url");
1215
1216 $header_array = array("package_name", "package_section", "class_name",
"class_full", "class_section", "method_name", "method_full", "method_para",
"version");
1217
1218 $scrape_file = "final7.csv";
1219
1220 if(!file_exists($scrape_file)){
1221
1222     $fp = fopen($scrape_file, 'a');
1223
1224     fputcsv($fp, $header_array);
1225
1226 }else{
1227
1228
1229     $fp = fopen($scrape_file, 'a');
1230
1231
1232 }
1233
1234
1235 $test = false;
1236
1237
1238 #check links for 'package-summary'
1239
1240 #$file_arr[] = 'C:/doc/1.0/doc/';
```

```
1241
1242
1243 //these are list of java doc for each J2SE version.
1244
1245 $file_arr[] = 'C:/doc/1.0/doc/';
1246 $file_arr[] = 'C:/doc/1.1/doc/';
1247 $file_arr[] = 'C:/doc/1.2/doc/';
1248 $file_arr[] = 'C:/doc/1.3/doc/';
1249 $file_arr[] = 'C:/doc/1.4/doc/';
1250 $file_arr[] = 'C:/doc/1.5/doc/';
1251 $file_arr[] = 'C:/doc/1.6/doc/';
1252 $file_arr[] = 'C:/doc/1.7/doc/';
1253 $file_arr[] = 'C:/doc/1.8/doc/';
1254
1255
1256
1257 /*
1258  * this section reads overview page from each java doc and calls methods
to get list of packages and its classes.
1259  */
1260
1261 for($j = 0; $j < sizeof($file_arr); $j++){
1262
1263     $url_arr = array();
1264     $url_arr = file_scrape($file_arr[$j].'overview-summary.html');
1265
1266     foreach($url_arr as $url){
1267
1268         $package_url = $url[0];
1269         $package_name = $url[1];
1270
1271         #var_dump(array($package_url, $package_name));
1272
1273         scrape($package_url, $package_name, $fp, $test, $file_arr[$j]);
1274
1275     }
1276
1277 }
1278
1279
1280
1281
1282 fclose($fp);
1283
1284
1285
1286
1287 ?>
1288 </pre>
```

APPENDIX F – PHP and R Scripts for Parsing XML Documentation of Android Lollipop SE API Packages

```
<pre>

<?php

/*
 *
 * The purpose of this script is to take text files detailing the
Android API packages, classes and methods
 *  and parse it into separate table detailing packages, classes,
methods with its type, access levels, and other info listed below
 *  and generate csvs for each table for spciefic android api level. It
is used to run on android API level 14-23.
 *
 *  Here is data dictionary of each table.
 *
 *  Package: name of the packages
 *
 *  Class: this table contains following columns
 *  package_name
 *  class_type
 *  class_name
 *  class_abstract
 *  class_static
 *  class_final
 *  class_deprecated
 *  class_visibility
 *  class_extends
 *  class_implements
 *
 *  Method: this table contains following columns
 *  package_name
 *  class_type
 *  class_name
 *  class_abstract
 *  class_static
 *  class_final
 *  class_deprecated
 *  class_visibility
 *  class_extends
 *  class_implements
 *  method_abstract
 *  method_synchronized
 *  method_exception
 *  method_return_type
 *  method_name
 *  method_transient
 *  method_volatile
 *  method_value
 *  method_parameter
 *  method_static
 *  method_final
 *  method_deprecated
 *  method_visibility
 *  method_type
 *
 *  This is sample of input data for api level 22
 *
```

```
 55  *  public final class Short extends java.lang.Number implements
java.lang.Comparable {
 56  *  ctor public Short(java.lang.String) throws
java.lang.NumberFormatException;
 57  *  ctor public Short(short);
 58  *  method public static int compare(short, short);
 59  *  method public int compareTo(java.lang.Short);
 60  *  method public static java.lang.Short decode(java.lang.String) throws
java.lang.NumberFormatException;
 61  *  field public static final int SIZE = 16; // 0x10
 62  *  field public static final java.lang.Class<java.lang.Short> TYPE;
 63  *
 64  *
 65  */
 66
 67
 68 /*
 69  *
 70  */
 71
 72 function parse($url){
 73
 74     $print_table = array();
 75
 76     $file_array = array();
 77
 78     /*
 79      * This Section reads lines from each file and create list/array for
all lines in file.
 80      */
 81     $handle = fopen($url, "r");
 82     if($handle){
 83         while(($buffer = fgets($handle, 4096)) !== false){
 84             array_push($file_array, $buffer);
 85         }
 86         if(!feof($handle)){
 87             echo "Error: unexpected fgets() fail\n";
 88         }
 89         fclose($handle);
 90     }
 91
 92
 93     $location = 0;
 94
 95     $this_arr = array();
 96
 97     $package_locations = array();
 98     $package_names = array();
 99
100     /*
101      * This Sections go through each line and parse it.
102      */
103
104     foreach($file_array as $file_row){
105
106         array_push($this_arr, substr($file_row, 0, 8));
107
```

```
108         /*
109          * This sections extracts package name from each line.
110          * It searches for "package" keyword in line,
111          * and if line has this keyword, it removes package keyword from
line and parse remaing text from line as package name.
112          *
113          * It tags line no whereever it finds package declration and
mark full package blocks containing classes.
114          * It creates list of all package names and put into package
table.
115          */
116
117         $package_len = strlen("package");
118
119         $package_part = substr($file_row, 0, $package_len);
120
121         if(strtolower($package_part) == 'package'){
122
123             $package = str_replace("package ", "", $file_row);
124             $this_package_name = str_replace(" {", "", $package);
125             array_push($package_names, $this_package_name);
126
127             array_push($print_table, array($this_package_name, $url));
128
129             array_push($package_locations, $location);
130
131         }
132
133         $location++;
134         #echo $first_part;
135
136     }
137
138     #print_r(array_unique($this_arr));
139
140     array_push($package_locations, $location);
141
142     #print_r($package_locations);
143
144     $this_package = array();
145
146     $final_package_table = array();
147     $final_class_table = array();
148     $final_method_table = array();
149
150
151
152     ### package loop
153
154     /*
155      * This sections goes through each package block and determines all
class blocks wihtin each package.
156      *
157      */
158
159     for($i = 0; $i < sizeof($package_locations)-1; $i++){
160
```

```
161         $package_name = $package_names[$i];
162
163         array_push($final_package_table, array($package_name, $url));
164
165         $start = $package_locations[$i]+1;
166         $end = $package_locations[$i+1];
167
168         for($j = $start; $j < $end; $j++){
169
170             array_push($this_package, $file_array[$j]);
171
172         }
173
174         $this_package_count = 0;
175
176         $class_locations_start = array();
177         $class_locations_end = array();
178
179         foreach($this_package as $package_row){
180
181             if(strpos($package_row, ' {')  !== false){
182
183                 array_push($class_locations_start, $this_package_count);
184
185             }
186
187             if(strpos($package_row, ' }')  !== false){
188
189                 array_push($class_locations_end, $this_package_count);
190
191             }
192
193             $this_package_count++;
194
195         }
196
197         #print_r($class_locations_end);
198
199
200         /*
201          * This sections goes through each class block examine each
section for
202          * class type, if class extends or implement, check if its
static or final etc. and create a class table with each field value.
203          *
204          */
205
206         ### class loop
207
208         for($k = 0; $k < sizeof($class_locations_start); $k++){
209
210             $start_cl = $class_locations_start[$k];
211             $end_cl = $class_locations_end[$k];
212
213
214
215             $this_class = array();
```

```
216
217             for($l = $start_cl; $l < $end_cl; $l++){
218
219                 array_push($this_class, $this_package[$l]);
220
221             }
222
223             if($k == 0){
224
225             #print_r($this_class);
226
227             }
228
229             #
230             $class_header_row = $this_class[0];
231             $class = str_replace(" {", "", $class_header_row);
232
233             $search_pos = -1;
234
235             $search_pos_count = 0;
236
237             if(strpos($class, " implements") !== false && strpos($class,
" extends") !== false){
238
239                 $search_pos = min(strpos($class, " implements"),
strpos($class, " extends"));
240                 $search_pos_count = 1;
241
242             }else{
243
244                 if(strpos($class, " implements") !== false){
245
246
247                     ### need to create file
248
249                     $search_pos = strpos($class, " implements");
250                     $search_pos_count = 2;
251
252                 }else{
253
254                     if(strpos($class, " extends") !== false){
255
256                         ## need to add to var
257
258                         $search_pos = strpos($class, " extends");
259                         $search_pos_count = 3;
260
261                     }
262
263                 }
264
265             }
266
267             $class_extends = "";
268
269             $class_extends_temp = "";
270
```

```
 271             $class_implements = "";
 272
 273             $class_implements_temp = "";
 274
 275             # public class SharedPreferencesBackupHelper extends
android.app.backup.FileBackupHelperBase implements
android.app.backup.BackupHelper {
 276
 277             if($search_pos > 0 && $search_pos_count == 1){
 278
 279                 ## extends first
 280                 if(strpos($class, " implements") > strpos($class, "
extends")){
 281
 282                     $class_extends_temp = substr($class, strpos($class,
" extends"), strpos($class, " implements") - strpos($class, " extends"));
 283
 284                     $class_extends_temp = explode(' ',
$class_extends_temp);
 285
 286                     $class_extends_piece_count = 0;
 287
 288                     foreach($class_extends_temp as
$class_extends_piece){
 289
 290                         if($class_extends_piece_count > 1){
 291
 292                             if(strlen($class_extends_piece) > 0){
 293
 294                                 if(strlen($class_extends) < 1){
 295
 296                                     $class_extends =
$class_extends_piece;
 297
 298                                 }else{
 299
 300                                     $class_extends = $class_extends.",
".$class_extends_piece;
 301
 302                                 }
 303
 304                             }
 305
 306                         }
 307
 308                         $class_extends_piece_count++;
 309
 310                     }
 311
 312                     $class_implements_temp = substr($class,
strpos($class, " implements"), strlen($class) - strpos($class, "
implements"));
 313
 314                     $class_implements_temp = explode(' ',
$class_implements_temp);
 315
 316                     $class_implements_piece_count = 0;
```

```
 317
 318                     foreach($class_implements_temp as
$class_implements_piece){
 319
 320                         if($class_implements_piece_count > 1){
 321
 322                             if(strlen($class_implements_piece) > 0 &&
strpos($class_implements_piece, '{') == false){
 323
 324                                 if(strlen($class_implements) < 1){
 325
 326                                     $class_implements =
$class_implements_piece;
 327
 328                                 }else{
 329
 330                                     $class_implements =
$class_implements.", ".$class_implements_piece;
 331
 332                                 }
 333
 334                             }
 335
 336                         }
 337
 338                         $class_implements_piece_count++;
 339
 340                     }
 341
 342
 343
 344                 }
 345
 346                 ## implements first
 347                 if(strpos($class, " implements") < strpos($class, "
extends")){
 348
 349                     $class_implements_temp = substr($class,
strpos($class, " implements"), strpos($class, " extends") - strpos($class, "
implements"));
 350
 351                     $class_implements_temp = explode(' ',
$class_implements_temp);
 352
 353                     $class_implements_piece_count = 0;
 354
 355                     foreach($class_implements_temp as
$class_implements_piece){
 356
 357                         if($class_implements_piece_count > 1){
 358
 359                             if(strlen($class_implements_piece) > 0){
 360
 361                                 if(strlen($class_implements) < 1){
 362
 363                                     $class_implements =
$class_implements_piece;
```

```
 364
 365                                    }else{
 366
 367                                        $class_implements =
$class_implements.", ".$class_implements_piece;
 368
 369                                    }
 370
 371                                }
 372
 373                            }
 374
 375                            $class_implements_piece_count++;
 376
 377                        }
 378
 379                        $class_extends_temp = substr($class, strpos($class,
" extends"), strlen($class) - strpos($class, " extends"));
 380
 381                        $class_extends_temp = explode(' ',
$class_extends_temp);
 382
 383                        $class_extends_piece_count = 0;
 384
 385                        foreach($class_extends_temp as
$class_extends_piece){
 386
 387                            if($class_extends_piece_count > 1){
 388
 389                                if(strlen($class_extends_piece) > 0 &&
strpos($class_extends_piece, '{') == false){
 390
 391                                    if(strlen($class_extends) < 1){
 392
 393                                        $class_extends =
$class_extends_piece;
 394
 395                                    }else{
 396
 397                                        $class_extends = $class_extends.",
".$class_extends_piece;
 398
 399                                    }
 400
 401                                }
 402
 403                            }
 404
 405                            $class_extends_piece_count++;
 406
 407                        }
 408
 409
 410
 411
 412                    }
 413
```

```
 414
 415             }
 416
 417             if($search_pos > 0 && $search_pos_count == 2){
 418
 419                 $class_implements_temp = substr($class, $search_pos,
strlen($class) - $search_pos);
 420
 421                 $class_implements_temp = explode(' ',
$class_implements_temp);
 422
 423                 $class_implements_piece_count = 0;
 424
 425                 foreach($class_implements_temp as
$class_implements_piece){
 426
 427                     if($class_implements_piece_count > 1){
 428
 429                         if(strlen($class_implements_piece) > 0 &&
strpos($class_implements_piece, '{') == false){
 430
 431                             if(strlen($class_implements) < 1){
 432
 433                                 $class_implements =
$class_implements_piece;
 434
 435                             }else{
 436
 437                                 $class_implements = $class_implements.",
".$class_implements_piece;
 438
 439                             }
 440
 441                         }
 442
 443                     }
 444
 445                     $class_implements_piece_count++;
 446
 447                 }
 448
 449             }
 450
 451             if($search_pos > 0 && $search_pos_count == 3){
 452
 453                 $class_extends_temp = substr($class, $search_pos,
strlen($class) - $search_pos);
 454
 455                 $class_extends_temp = explode(' ', $class_extends_temp);
 456
 457                 $class_extends_piece_count = 0;
 458
 459                 foreach($class_extends_temp as $class_extends_piece){
 460
 461                     if($class_extends_piece_count > 1){
 462
```

```
463                         if(strlen($class_extends_piece) > 0 &&
strpos($class_extends_piece, '{') == false){
464
465                             if(strlen($class_extends) < 1){
466
467                                 $class_extends = $class_extends_piece;
468
469                             }else{
470
471                                 $class_extends = $class_extends.",
".$class_extends_piece;
472
473                             }
474
475                         }
476
477                     }
478
479                     $class_extends_piece_count++;
480
481                 }
482
483             }
484
485             #echo "extends ". $class_extends . " implements " .
$class_implements . "\n\r";
486
487
488
489
490             if($search_pos > 0){
491
492                 $class = substr($class, 0, $search_pos);
493
494             }
495
496             #interface, class
497             $class_type = "";
498             $class_name = '';
499             $class_abstract = 0;
500             $class_static = 0;
501             $class_final = 0;
502             $class_deprecated = 0;
503             $class_visibility = "";
504
505             if(strpos($class, "class ") !== false){
506
507                 $class_type = "class";
508
509             }else{
510
511                 if(strpos($class, "interface ") !== false){
512
513                     $class_type = "interface";
514
515                 }
516
```

```
517             }
518
519             if(strpos($class, "abstract ") !== false){
520
521                 $class_abstract = 1;
522
523             }
524
525             if(strpos($class, "static ") !== false){
526
527                 $class_static = 1;
528
529             }
530
531             if(strpos($class, "final ") !== false){
532
533                 $class_final = 1;
534
535             }
536
537             if(strpos($class, "deprecated ") !== false){
538
539                 $class_deprecated = 1;
540
541             }
542
543
544             if(strpos($class, "public ") !== false){
545
546                 $class_visibility = "public";
547
548             }else{
549
550                 if(strpos($class, "protected ") !== false){
551
552                 $class_visibility = "protected";
553
554                 }else{
555
556                     if(strpos($class, "private ") !== false){
557
558                     $class_visibility = "private";
559
560                     }else{
561
562                         $class_visibility = "no modifier";
563
564                     }
565
566                 }
567
568             }
569
570
571
572
573
```

```
574
575             #var_dump($class);
576
577             $class = explode(' ', $class);
578
579             #var_dump($class);
580
581             $class_name = $class[sizeof($class)-1];
582
583             $dump = $package_name."@".$class_name."-".$url;
584
585             $dump = trim(preg_replace('/\s\s+/', ' ', $dump));
586
587             $dump = str_replace("n\\", '', $dump);
588             $dump = str_replace("r\\", '', $dump);
589             $dump = str_replace("t\\", '', $dump);
590             $dump = str_replace("n\r\\", '', $dump);
591             $dump = str_replace("r\n\\", '', $dump);
592             $dump = preg_replace('~[\r\n]+~', '', $dump);
593
594
595             /*
596              * this section identifies method blocks from each class
block and create list of these method blocks to process.
597              */
598             #echo $dump."\n\r";
599
600             $method_text_array = array();
601
602             $header_count = 0;
603
604             foreach($this_class as $this_class_row){
605
606                 if($header_count > 0){
607
608                     if(strlen($this_class_row) > 2){
609
610                         array_push($method_text_array, $this_class_row);
611
612                     }
613
614                 }
615
616                 $header_count++;
617
618             }
619
620             $final_class_array = array($package_name, $class_type,
$class_name, $class_abstract, $class_static, $class_final, $class_deprecated,
$class_visibility, $class_extends, $class_implements, $url);
621
622             array_push($final_class_table, $final_class_array);
623
624
625             /*
626              * This sections processes each method blocks and identifies
if it is method, field or constructor.
```

```
627              */
628
629             ### method loop
630
631             foreach($method_text_array as $method_text){
632
633                 $method_type = "";
634
635                 #ctor, field, method, $enum_constant
636                 $method_explode = explode(' ', $method_text);
637
638                 $method_turned = 0;
639
640                 $method_first_piece = "";
641
642                 foreach($method_explode as $method_piece){
643
644                     if(strlen($method_piece) > 0 && $method_turned ==
645 0){
646
647                         $method_first_piece = $method_piece;
648                         $method_turned = 1;
649
650                     }
651
652                 }
653
654                 if(strpos($method_first_piece, "enum_constant") !==
655 false){
656
657                     $method_type = "enumerator value";
658
659                 }else{
660
661                     if(strpos($method_first_piece, "method") !== false){
662
663                     $method_type = "method";
664
665                     }else{
666
667                         if(strpos($method_first_piece, "field") !==
668 false){
669
670                         $method_type = "field";
671
672                         }else{
673
674                             if(strpos($method_first_piece, "ctor") !==
675 false){
676
677                             $method_type = "constructor";
678
679                             }else{
680
681                                 $method_type = "could not locate method
682 type";
683
```

```
679                             }
680
681                         }
682
683                     }
684
685                 }
686
687
688                 /*
689                  * This sections processes each method blocks and
identifies type of method and its access scope.
690                  */
691
692                 #echo $method_type."\n\r";
693
694
695                 $method_static = 0;
696                 $method_final = 0;
697                 $method_deprecated = 0;
698                 $method_visibility = "";
699
700                 if(strpos($method_text, "static ") !== false){
701
702                     $method_static = 1;
703
704                 }
705
706                 if(strpos($method_text, "final ") !== false){
707
708                     $method_final = 1;
709
710                 }
711
712                 if(strpos($method_text, "deprecated ") !== false){
713
714                     $method_deprecated = 1;
715
716                 }
717
718                 if(strpos($method_text, "public ") !== false){
719
720                 $method_visibility = "public";
721
722                 }else{
723
724                     if(strpos($method_text, "protected ") !== false){
725
726                     $method_visibility = "protected";
727
728                     }else{
729
730                         if(strpos($method_text, "private ") !== false){
731
732                         $method_visibility = "private";
733
734                         }else{
```

```
 735
 736                             $method_visibility = "no modifier";
 737
 738                         }
 739
 740                     }
 741
 742                 }
 743
 744                 /*
 745                  * This sections processes each method blocks and
identifies field type and its access level.
 746                  */
 747
 748                 $method_return_type = '';
 749                 $method_name = '';
 750                 $method_transient = '';
 751                 $method_volatile = '';
 752                 $method_value = '';
 753                 $method_parameter = '';
 754
 755                 if($method_type == 'field'){
 756
 757                     $method_text_temp = $method_text;
 758
 759                     $method_text_temp = explode(' ', $method_text_temp);
 760
 761                     $method_text_count = 0;
 762                     $method_count_match = 0;
 763
 764                     foreach($method_text_temp as $method_temp_piece){
 765
 766                         if(strpos(strtolower($method_temp_piece), "=")
!== false && $method_count_match == 0){
 767
 768                             $method_count_match = $method_text_count;
 769
 770                             #echo $method_temp_piece;
 771
 772                         }
 773
 774                         $method_text_count++;
 775
 776                     }
 777
 778                     if($method_count_match >= 2){
 779
 780                         $method_return_type =
$method_text_temp[$method_count_match-2];
 781                         $method_name =
$method_text_temp[$method_count_match-1];
 782                         $method_value = str_replace(';', '',
$method_text_temp[$method_count_match+1]);
 783
 784                     }else{
 785
```

```
 786                         $method_name = str_replace(';', '',
$method_text_temp[sizeof($method_text_temp)-1]);
 787                         $method_return_type = str_replace(';', '',
$method_text_temp[sizeof($method_text_temp)-2]);
 788
 789                     }
 790
 791                     if(strpos($method_text, "transient ") !== false){
 792
 793                         $method_transient = 1;
 794
 795                     }else{
 796
 797                         $method_transient = 0;
 798
 799                     }
 800
 801                     if(strpos($method_text, "volatile ") !== false){
 802
 803                         $method_volatile = 1;
 804
 805                     }else{
 806
 807                         $method_volatile = 0;
 808
 809                     }
 810
 811                     #echo $method_name;
 812
 813                 }
 814
 815                 /*
 816                  * This sections processes each method blocks and
extract method name and check if it is abstract or synchronized.
 817                  */
 818
 819                 $method_abstract = "";
 820                 $method_synchronized = '';
 821                 $method_exception = '';
 822
 823                 if($method_type == 'method'){
 824
 825                     $method_text_temp = $method_text;
 826
 827                     $method_text_temp = explode(' ', $method_text_temp);
 828
 829                     $method_text_count = 0;
 830                     $method_count_match = 0;
 831
 832                     foreach($method_text_temp as $method_temp_piece){
 833
 834                         if(strpos(strtolower($method_temp_piece), "(")
!== false && $method_count_match == 0){
 835
 836                             $method_count_match = $method_text_count;
 837
 838                             #echo $method_temp_piece;
```

```
 839
 840                         }
 841
 842                         $method_text_count++;
 843
 844                     }
 845
 846                     $method_return_type =
$method_text_temp[$method_count_match-1];
 847                     $method_name_temp =
$method_text_temp[$method_count_match];
 848
 849                     $method_name_temp = explode('(', $method_name_temp);
 850                     $method_name = $method_name_temp[0];
 851
 852                     $method_parameter_temp = explode(')',
$method_name_temp[1]);
 853                     $method_parameter = $method_parameter_temp[0];
 854                     #echo $method_parameter;
 855                     #echo $method_return_type;
 856
 857                     if(strpos($method_text, "abstract ") !== false){
 858
 859                         $method_abstract = 1;
 860
 861                     }else{
 862
 863                         $method_abstract = 0;
 864
 865                     }
 866
 867                     if(strpos($method_text, "synchronized ") !== false){
 868
 869                         $method_synchronized = 1;
 870
 871                     }else{
 872
 873                         $method_synchronized = 0;
 874
 875                     }
 876
 877
 878                     if(strpos($method_text, " throws ") !== false){
 879
 880                         $throws_pos = strpos($method_text, " throws ");
 881
 882                         if($throws_pos > 0){
 883
 884                             $method_throws_temp = substr($method_text,
$throws_pos, strlen($method_text) - $throws_pos);
 885
 886                             #$method_throws_temp = explode(' ',
$method_throws_temp);
 887
 888                             #$method_exception = str_replace(';', '',
$method_throws_temp[sizeof($method_throws_temp)-1]);
 889
```

```
 890                                #echo $method_exception;
 891
 892                                $method_throws_temp = str_replace(' throws
', '', $method_throws_temp);
 893
 894                                $method_exception = str_replace(';', '',
$method_throws_temp);
 895
 896                            }
 897
 898                        }
 899
 900
 901
 902                }
 903
 904                /*
 905                 * This sections processes each method blocks and
identifies constructor related information.
 906                 */
 907
 908                if($method_type == 'constructor'){
 909
 910                    $c_method_text_temp = $method_text;
 911
 912                    $c_method_text_temp = explode(' ',
$c_method_text_temp);
 913
 914                    $c_method_text_count = 0;
 915                    $c_method_count_match = 0;
 916
 917                    foreach($c_method_text_temp as
$c_method_temp_piece){
 918
 919                        if(strpos(strtolower($c_method_temp_piece), "(")
!== false && $c_method_count_match == 0){
 920
 921                            $c_method_count_match =
$c_method_text_count;
 922
 923                            #echo $method_temp_piece;
 924
 925                        }
 926
 927                        $c_method_text_count++;
 928
 929                    }
 930
 931                    $c_method_name_temp =
$c_method_text_temp[$c_method_count_match];
 932                    $c_method_name_temp = explode('(',
$c_method_name_temp);
 933                    $method_name = $c_method_name_temp[0];
 934
 935                    $c_method_parameter_temp = explode(')',
$c_method_name_temp[1]);
 936                    $method_parameter = $c_method_parameter_temp[0];
```

```
 937
 938                        #echo $method_parameter;
 939
 940
 941                        $method_return_type =
$package_name."."$method_name;
 942                        #echo $method_return_type;
 943
 944                        if(strpos($method_text, " throws ") !== false){
 945
 946                            $c_throws_pos = strpos($method_text, " throws
");
 947
 948                            if($c_throws_pos > 0){
 949
 950                                $c_method_throws_temp = substr($method_text,
$c_throws_pos, strlen($method_text) - $c_throws_pos);
 951
 952                                $c_method_throws_temp = str_replace(' throws
', '', $c_method_throws_temp);
 953
 954                                $method_exception = str_replace(';', '',
$c_method_throws_temp);
 955
 956                                #$method_exception = str_replace(';', '',
$c_method_throws_temp[sizeof($c_method_throws_temp)-1]);
 957
 958                                #echo $method_exception;
 959
 960                            }
 961
 962                        }
 963
 964
 965                    }
 966
 967
 968                    $final_method_array = array($package_name, $class_type,
$class_name, $class_abstract, $class_static, $class_final, $class_deprecated,
$class_visibility, $class_extends, $class_implements, $method_abstract,
$method_synchronized, $method_exception, $method_return_type, $method_name,
$method_transient, $method_volatile, $method_value, $method_parameter,
$method_static, $method_final, $method_deprecated, $method_visibility,
$method_type, $url);
 969
 970                    if(!($method_type == 'enumerator value')){
 971
 972                        array_push($final_method_table,
$final_method_array);
 973
 974                    }
 975
 976
 977                }
 978
 979
 980                #$class_type =
```

```
981
982             #echo $class;
983
984             #var_dump($class);
985
986
987
988             #
989
990
991         }
992
993
994
995         $this_package = array();
996
997     }
998
999     /*
1000
1001    foreach($print_table as $print_this_row){
1002
1003        fputcsv($fp, $print_this_row);
1004
1005    }
1006
1007    */
1008
1009
1010    #$url_array = array();
1011
1012    /*array_push($url_array, 'C:\android-source-doc\3');
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027    #print_r($result_table);
1028
1029    fclose($fp);
1030
1031    $finish_file = fopen("finish.txt", "w");
1032    fwrite($finish_file, "done.");
1033    fclose($finish_file);
1034
1035    */
1036
```

```
1037     return array($final_package_table, $final_class_table,
$final_method_table);
1038
1039 }
1040
1041 /*
1042  * This section reads input directory and create list of input files
needs to be processes.
1043  */
1044 $local_dir = 'C:\Users\azvorinji\Documents\android api\\';
1045
1046 $files = scandir($local_dir);
1047
1048 $url_list = array();
1049
1050 foreach($files as $file){
1051
1052     if(strpos($file, ".txt") !== false){
1053
1054         array_push($url_list, $local_dir.$file);
1055
1056     }
1057
1058 }
1059
1060 #$scrape_file = 'C:\Users\azvorinji\Documents\android api\4.0.1_r1.txt';
1061 #$scrape_file = 'C:\Users\azvorinji\Documents\android api\6.0.0_r1.txt';
1062 #$text_file_content = file_get_contents($scrape_file);
1063
1064 /*
1065  * This section create new CSV file for package details and initialize
it with column headers.
1066  */
1067
1068 $package_header_array = array("package_name", "version");
1069
1070 $package_scrape_file = 'package_3.csv';
1071
1072 if(!file_exists($package_scrape_file)){
1073
1074     $fp = fopen($package_scrape_file, 'a');
1075
1076     fputcsv($fp, $package_header_array);
1077
1078 }else{
1079
1080
1081     $fp = fopen($package_scrape_file, 'a');
1082
1083
1084 }
1085
1086
1087 /*
1088  * This section create new CSV file for Class details and initialize it
with column headers.
1089  */
```

```
1090
1091
1092 $class_header_array = array("package_name", "class_type", "class_name",
"class_abstract", "class_static", "class_final", "class_deprecated",
"class_visibility", "class_extends", "class_implements", "version");
1093
1094 $class_scrape_file = 'class_3.csv';
1095
1096 if(!file_exists($class_scrape_file)){
1097
1098     $fc = fopen($class_scrape_file, 'a');
1099
1100     fputcsv($fc, $class_header_array);
1101
1102 }else{
1103
1104
1105     $fc = fopen($class_scrape_file, 'a');
1106
1107
1108 }
1109
1110
1111 /*
1112  * This section create new CSV file for methods detailing and initialize
it with column headers.
1113  */
1114
1115
1116 $method_header_array = array("package_name", "class_type", "class_name",
"class_abstract", "class_static", "class_final", "class_deprecated",
"class_visibility", "class_extends", "class_implements", "method_abstract",
"method_synchronized", "method_exception", "method_return_type",
"method_name", "method_transient", "method_volatile", "method_value",
"method_parameter", "method_static", "method_final", "method_deprecated",
"method_visibility", "method_type", "version");
1117
1118 $method_scrape_file = 'method_3_14.csv';
1119
1120 if(!file_exists($method_scrape_file)){
1121
1122     $fm = fopen($method_scrape_file, 'a');
1123
1124     fputcsv($fm, $method_header_array);
1125
1126 }else{
1127
1128
1129     $fm = fopen($method_scrape_file, 'a');
1130
1131
1132 }
1133
1134
1135 /*
1136  * This section reads each input files and call parse API to generate 3
different tables.
```

```
1137   * 1. packages
1138   * 2. classes
1139   * 3. methods
1140   *
1141   * Then it processes each table into CSV format and write each row into
CSV files.
1142   */
1143
1144 foreach($url_list as $url){
1145
1146
1147     $return_table = parse($url);
1148
1149     $package_table = $return_table[0];
1150     $class_table = $return_table[1];
1151     $method_table = $return_table[2];
1152
1153     $print_table_package = array();
1154     $print_table_class = array();
1155     $print_table_method = array();
1156
1157     $dump = '';
1158
1159     foreach($package_table as $package_array){
1160
1161         $print_array_package = array();
1162
1163         foreach($package_array as $package_value){
1164
1165             $dump = $package_value;
1166
1167             $dump = trim(preg_replace('/\s\s+/', ' ', $dump));
1168             $dump = str_replace("n\\", '', $dump);
1169             $dump = str_replace("r\\", '', $dump);
1170             $dump = str_replace("t\\", '', $dump);
1171             $dump = str_replace("n\r\\", '', $dump);
1172             $dump = str_replace("r\n\\", '', $dump);
1173             $dump = preg_replace('~[\r\n]+~', '', $dump);
1174
1175             $package_value = $dump;
1176
1177             array_push($print_array_package, $package_value);
1178
1179         }
1180
1181         array_push($print_table_package, $print_array_package);
1182
1183     }
1184
1185     foreach($class_table as $class_array){
1186
1187         $print_array_class = array();
1188
1189         foreach($class_array as $class_value){
1190
1191             $dump = $class_value;
1192             $dump = trim(preg_replace('/\s\s+/', ' ', $dump));
```

```
1193            $dump = str_replace("n\\", '', $dump);
1194            $dump = str_replace("r\\", '', $dump);
1195            $dump = str_replace("t\\", '', $dump);
1196            $dump = str_replace("n\r\\", '', $dump);
1197            $dump = str_replace("r\n\\", '', $dump);
1198            $dump = preg_replace('~[\r\n]+~', '', $dump);
1199
1200            $class_value = $dump;
1201
1202            array_push($print_array_class, $class_value);
1203
1204        }
1205
1206        array_push($print_table_class, $print_array_class);
1207
1208    }
1209
1210    foreach($method_table as $method_array){
1211
1212        $print_array_method = array();
1213
1214        foreach($method_array as $method_value){
1215
1216            $dump = $method_value;
1217            $dump = trim(preg_replace('/\s\s+/', ' ', $dump));
1218            $dump = str_replace("n\\", '', $dump);
1219            $dump = str_replace("r\\", '', $dump);
1220            $dump = str_replace("t\\", '', $dump);
1221            $dump = str_replace("n\r\\", '', $dump);
1222            $dump = str_replace("r\n\\", '', $dump);
1223            $dump = preg_replace('~[\r\n]+~', '', $dump);
1224
1225            $method_value = $dump;
1226
1227            array_push($print_array_method, $method_value);
1228
1229        }
1230
1231        array_push($print_table_method, $print_array_method);
1232
1233    }
1234
1235    foreach($print_table_package as $print_this_row){
1236
1237        fputcsv($fp, $print_this_row);
1238
1239    }
1240
1241    foreach($print_table_class as $print_this_row){
1242
1243        fputcsv($fc, $print_this_row);
1244
1245    }
1246
1247    foreach($print_table_method as $print_this_row){
1248
1249        fputcsv($fm, $print_this_row);
```

```
1250
1251     }
1252
1253
1254
1255 }
1256
1257 $finish_file = fopen("finish.txt", "w");
1258 fwrite($finish_file, "done.");
1259 fclose($finish_file);
1260
1261 ?>
1262 </pre>
```

```
  1 #install related packages
  2 install.packages("plyr")
  3 install.packages("tidyr")
  4 library(plyr)
  5
  6 #pull in and clean the data files
  7 setwd("C:/Users/mshen/Box Sync/Oracle-Google/workstreams/22_API
popularity analysis/android evolution/android api/3.2.4_r1")
  8
  9 package <- read.table("package.csv", header = TRUE, sep = ",",
stringsAsFactors = FALSE)
 10 colnames(package)[1] <- "package_Id"
 11 colnames(package)[2] <- "package_name"
 12
 13 class <- read.table("class.csv", header = TRUE, sep = ",",
stringsAsFactors = FALSE)
 14 colnames(class)[1] <- "class_id"
 15 class$class_id <- paste("class", class$class_id)
 16 colnames(class)[2] <- "class_name"
 17 colnames(class)[3] <- "class_extends"
 18 colnames(class)[4] <- "class_abstract"
 19 colnames(class)[5] <- "class_static"
 20 colnames(class)[6] <- "class_final"
 21 colnames(class)[7] <- "class_deprecated"
 22 colnames(class)[8] <- "class_visibility"
 23 class$class_type <- "class"
 24 class <- merge(x = package, y = class, by = "package_Id", all.y = TRUE)
 25
 26 interface <- read.table("interface.csv", header = TRUE, sep = ",",
stringsAsFactors = FALSE)
 27 colnames(interface)[1] <- "class_id"
 28 interface$class_id <- paste("interface", interface$class_id)
 29 colnames(interface)[2] <- "class_name"
 30 colnames(interface)[3] <- "class_abstract"
 31 colnames(interface)[4] <- "class_static"
 32 colnames(interface)[5] <- "class_final"
 33 colnames(interface)[6] <- "class_deprecated"
 34 colnames(interface)[7] <- "class_visibility"
 35 interface$class_type <- "interface"
 36 interface$class_extends <- ""
 37 interface <- interface[,c(1,2,10,3,4,5,6,7,8,9)]
```

```
 38 interface <- merge(x = package, y = interface, by = "package_Id", all.y =
TRUE)
 39
 40 #combine class and interface files
 41 final_class <- rbind(class,interface)
 42
 43 # join implements with class
 44
 45 implements <- read.table("implements.csv", header = TRUE, sep = ",",
stringsAsFactors = FALSE)
 46 colnames(implements)[1] <- "class_implements"
 47 implements$class_Id <- ifelse(is.na(implements$class_Id),"",
paste("class",implements$class_Id))
 48 implements$interface_Id <- ifelse(is.na(implements$interface_Id),"",
paste("interface",implements$interface_Id))
 49 implements$class_Inter_id <-
paste(implements$class_Id,implements$interface_Id)
 50 trim <- function (x) gsub("^\\s+|\\s+$", "", x)
 51 implements$class_Inter_id <- trim(implements$class_Inter_id)
 52 implements$class_Id <- NULL
 53 implements$interface_Id <- NULL
 54 colnames(implements)[2] <- "class_id"
 55 implements <- aggregate(class_implements~class_id, paste,collapse = ",",
data = implements)
 56
 57 final_class <- merge(x = final_class, y = implements, by = "class_id",
all = TRUE)
 58 final_class <- final_class[,c(2,1,3,11,4,6,7,8,9,10,5,12)]
 59 final_class$version <- "3.2.4"
 60
 61 # pull in method, constructor and field datasets
 62
 63 construct <- read.table("constructor.csv", header = TRUE, sep = ",",
stringsAsFactors = FALSE)
 64 colnames(construct)[1] <- "Id"
 65 construct$Id <- paste("construct",construct$Id)
 66 colnames(construct)[8] <- "class_Inter_id"
 67 construct$class_Inter_id <- paste("class",construct$class_Inter_id)
 68 construct$transient <- NA
 69 construct$volatile <- NA
 70 construct$value <- NA
 71 construct$return <- NA
 72 construct$abstract <- NA
 73 construct$native <- NA
 74 construct$synchronized <- NA
 75 construct <- construct[,c(1,2,3,9,10,11,12,13,14,15,4,5,6,7,8)]
 76 construct$category <- "constructor"
 77
 78 field <- read.table("field.csv", header = TRUE, sep = ",",
stringsAsFactors = FALSE)
 79 field$class_Id <- ifelse(is.na(field$class_Id),"",
paste("class",field$class_Id))
 80 field$interface_Id <- ifelse(is.na(field$interface_Id),"",
paste("interface",field$interface_Id))
 81 field$class_Inter_id <- paste(field$class_Id,field$interface_Id)
 82 field$class_Id <- NULL
 83 field$interface_Id <- NULL
```

```
 84 field$Id <- field$Id <- seq(1:nrow(field))
 85 field <- field[,c(11,1,2,3,4,5,6,7,8,9,10)]
 86 field$Id <- paste("field",field$Id)
 87 field$return <- NA
 88 field$abstract <- NA
 89 field$native <- NA
 90 field$synchronized <- NA
 91 field <- field[,c(1,2,3,4,5,6,12,13,14,15,7,8,9,10,11)]
 92 colnames(field)[2] <- "name"
 93 trim <- function (x) gsub("^\\s+|\\s+$", "", x)
 94 field$class_Inter_id <- trim(field$class_Inter_id)
 95 field$category <- "field"
 96
 97 method <- read.table("method.csv", header = TRUE, sep = ",",
stringsAsFactors = FALSE)
 98 colnames(method)[1] <- "Id"
 99 method$Id <- paste("method",method$Id)
100 method$class_Id <- ifelse(is.na(method$class_Id),"",
paste("class",method$class_Id))
101 method$interface_Id <- ifelse(is.na(method$interface_Id),"",
paste("interface",method$interface_Id))
102 method$class_Inter_id <- paste(method$class_Id,method$interface_Id)
103 method$class_Id <- NULL
104 method$interface_Id <- NULL
105 method$type <- NA
106 method$transient <- NA
107 method$volatile <- NA
108 method$value <- NA
109 method <- method[,c(1,2,12,13,14,15,3,4,5,6,7,8,9,10,11)]
110 trim <- function (x) gsub("^\\s+|\\s+$", "", x)
111 method$class_Inter_id <- trim(method$class_Inter_id)
112 method$category <- "method"
113
114 #combine constructor, filed and method
115 method_all <- rbind(method, field, construct)
116 colnames(method_all) <-
c("method_id","method_name","method_type","method_transient","method_volatile
","method_value","method_return","method_abstract","method_native","method_sy
nchronized","method_static","method_final","method_deprecated","method_visibi
lity","class_id","method_category")
117
118 #join method_all with exception and parameter
119 exception <- read.table("exception.csv", header = TRUE, sep = ",",
stringsAsFactors = FALSE)
120 exception$method_Id <- ifelse(is.na(exception$method_Id),"",
paste("method",exception$method_Id))
121 exception$constructor_Id <- ifelse(is.na(exception$constructor_Id),"",
paste("construct",exception$constructor_Id))
122 exception$Id <- paste(exception$method_Id,exception$constructor_Id)
123 exception$method_Id <- NULL
124 exception$constructor_Id <- NULL
125 trim <- function (x) gsub("^\\s+|\\s+$", "", x)
126 exception$Id <- trim(exception$Id)
127 colnames(exception)[1] <- "name"
128 exception$type_name <- paste(exception$type," ",exception$name)
129 exception$name <- NULL
130 exception1 <- aggregate(type~Id, paste,collapse = ",", data = exception)
```

```
131 exception2 <- aggregate(type_name~Id, paste,collapse = ",", data =
exception)
132 exception <- cbind(exception1,exception2)
133 exception <- exception[,-1]
134 colnames(exception)[1] <- "exception_type"
135 colnames(exception)[3] <- "exception_type_name"
136 colnames(exception)[2] <- "method_id"
137
138 parameter <- read.table("parameter.csv", header = TRUE, sep = ",",
stringsAsFactors = FALSE)
139 parameter$method_Id <- ifelse(is.na(parameter$method_Id),"",
paste("method",parameter$method_Id))
140 parameter$constructor_Id <- ifelse(is.na(parameter$constructor_Id),"",
paste("construct",parameter$constructor_Id))
141 parameter$Id <- paste(parameter$method_Id,parameter$constructor_Id)
142 parameter$method_Id <- NULL
143 parameter$constructor_Id <- NULL
144 trim <- function (x) gsub("^\\s+|\\s+$", "", x)
145 parameter$Id <- trim(parameter$Id)
146 parameter$type_name <- paste(parameter$type," ",parameter$ï..name)
147 parameter$ï..name <- NULL
148 parameter1 <- aggregate(type~Id, paste,collapse = ",", data = parameter)
149 parameter2 <- aggregate(type_name~Id, paste,collapse = ",", data =
parameter)
150 parameter <- cbind(parameter1,parameter2)
151 parameter <- parameter[,-3]
152 colnames(parameter)[2] <- "parameter_type"
153 colnames(parameter)[3] <- "parameter_type_name"
154 colnames(parameter)[1] <- "method_id"
155
156 method_final <- join(method_all,exception, by = "method_id",type =
"full", match = "all")
157 method_final <- join(method_final, parameter, by = "method_id", type =
"full", match = "all")
158
159 #merge method_final with final_class
160 final_method <- join(final_class, method_final, by = "class_id", type =
"full", match = "all")
161 final_method <-
final_method[,c(3,4,5,6,7,8,9,10,11,12,21,23,29,16,20,15,17,18,19,32,24,25,26
,27,28,13)]
162
163 final_method$class_abstract <- ifelse(final_method$class_abstract ==
"true",1,0)
164 final_method$class_static <- ifelse(final_method$class_static ==
"true",1,0)
165 final_method$class_final <- ifelse(final_method$class_final ==
"true",1,0)
166 final_method$class_deprecated <- ifelse(final_method$class_deprecated ==
"deprecated",1,0)
167 final_method[is.na(final_method)] <- ""
168 final_method$method_abstract <-
ifelse(final_method$method_abstract=="false",0,ifelse(final_method$method_abs
tract=="true",1,""))
169 final_method$method_synchronized <-
ifelse(final_method$method_synchronized=="false",0,ifelse(final_method$method
_synchronized=="true",1,""))
```

```
colnames(final_method)[13] <- "method_exception"
final_method$method_return_type <-
paste(final_method$method_type,final_method$method_return)
final_method <-
final_method[,c(1,2,3,4,5,6,7,8,9,10,11,12,13,27,16,17,18,19,20,21,22,23,24,25,26)]
final_method$method_transient <-
ifelse(final_method$method_transient=="false",0,ifelse(final_method$method_transient=="true",1,""))
final_method$method_volatile <-
ifelse(final_method$method_volatile=="false",0,ifelse(final_method$method_volatile=="true",1,""))
colnames(final_method)[19] <- "method_parameter"
final_method$method_static <- ifelse(final_method$method_static ==
"true",1,0)
final_method$method_final <- ifelse(final_method$method_final ==
"true",1,0)
final_method$method_deprecated <- ifelse(final_method$method_deprecated
== "deprecated",1,0)
colnames(final_method)[24] <- "method_type"

final_class <- final_method[,c(1,2,3,4,5,6,7,8,9,10,25)]
package$version <- "3.2.4"
final_package <- package[,c(2,3)]

#save to csv file
write.table(final_package, "C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_package3.2.4.csv", sep = ",", row.names= FALSE)
write.table(final_class, "C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_class3.2.4.csv", sep = ",", row.names= FALSE)
write.table(final_method, "C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_method3.2.4.csv", sep = ",", row.names= FALSE)

########### above code changed becasue of method exception in the final
data only needs method_type###
#read all_versions
package1.0 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_package1.0.csv", header = TRUE, sep = ",", stringsAsFactors
= FALSE)
class1.0 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_class1.0.csv", header = TRUE, sep = ",", stringsAsFactors =
FALSE)
method1.0 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_method1.0.csv", header = TRUE, sep = ",", stringsAsFactors
= FALSE)

package1.1 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_package1.1.csv", header = TRUE, sep = ",", stringsAsFactors
= FALSE)
```

```
197 class1.1 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_class1.1.csv", header = TRUE, sep = ",", stringsAsFactors =
FALSE)
198 method1.1 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_method1.1.csv", header = TRUE, sep = ",", stringsAsFactors
= FALSE)
199
200 package1.5 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_package1.5.csv", header = TRUE, sep = ",", stringsAsFactors
= FALSE)
201 class1.5 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_class1.5.csv", header = TRUE, sep = ",", stringsAsFactors =
FALSE)
202 method1.5 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_method1.5.csv", header = TRUE, sep = ",", stringsAsFactors
= FALSE)
203
204 package1.6 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_package1.6.csv", header = TRUE, sep = ",", stringsAsFactors
= FALSE)
205 class1.6 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_class1.6.csv", header = TRUE, sep = ",", stringsAsFactors =
FALSE)
206 method1.6 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_method1.6.csv", header = TRUE, sep = ",", stringsAsFactors
= FALSE)
207
208 package2.0.1 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_package2.0.1.csv", header = TRUE, sep = ",",
stringsAsFactors = FALSE)
209 class2.0.1 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_class2.0.1.csv", header = TRUE, sep = ",", stringsAsFactors
= FALSE)
210 method2.0.1 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_method2.0.1.csv", header = TRUE, sep = ",",
stringsAsFactors = FALSE)
211
212 package2.0 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_package2.0.csv", header = TRUE, sep = ",", stringsAsFactors
= FALSE)
213 class2.0 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_class2.0.csv", header = TRUE, sep = ",", stringsAsFactors =
FALSE)
```

```
214 method2.0 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_method2.0.csv", header = TRUE, sep = ",", stringsAsFactors
= FALSE)
215
216 package2.1 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_package2.1.csv", header = TRUE, sep = ",", stringsAsFactors
= FALSE)
217 class2.1 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_class2.1.csv", header = TRUE, sep = ",", stringsAsFactors =
FALSE)
218 method2.1 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_method2.1.csv", header = TRUE, sep = ",", stringsAsFactors
= FALSE)
219
220 package2.2 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_package2.2.csv", header = TRUE, sep = ",", stringsAsFactors
= FALSE)
221 class2.2 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_class2.2.csv", header = TRUE, sep = ",", stringsAsFactors =
FALSE)
222 method2.2 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_method2.2.csv", header = TRUE, sep = ",", stringsAsFactors
= FALSE)
223
224 package2.3.3 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_package2.3.3.csv", header = TRUE, sep = ",",
stringsAsFactors = FALSE)
225 class2.3.3 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_class2.3.3.csv", header = TRUE, sep = ",", stringsAsFactors
= FALSE)
226 method2.3.3 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_method2.3.3.csv", header = TRUE, sep = ",",
stringsAsFactors = FALSE)
227
228 package2.3 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_package2.3.csv", header = TRUE, sep = ",", stringsAsFactors
= FALSE)
229 class2.3 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_class2.3.csv", header = TRUE, sep = ",", stringsAsFactors =
FALSE)
230 method2.3 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_method2.3.csv", header = TRUE, sep = ",", stringsAsFactors
= FALSE)
231
```

```
232 package3.0 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_package3.0.csv", header = TRUE, sep = ",", stringsAsFactors
= FALSE)
233 class3.0 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_class3.0.csv", header = TRUE, sep = ",", stringsAsFactors =
FALSE)
234 method3.0 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_method3.0.csv", header = TRUE, sep = ",", stringsAsFactors
= FALSE)
235
236 package3.1 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_package3.1.csv", header = TRUE, sep = ",", stringsAsFactors
= FALSE)
237 class3.1 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_class3.1.csv", header = TRUE, sep = ",", stringsAsFactors =
FALSE)
238 method3.1 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_method3.1.csv", header = TRUE, sep = ",", stringsAsFactors
= FALSE)
239
240 package3.2.4 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_package3.2.4.csv", header = TRUE, sep = ",",
stringsAsFactors = FALSE)
241 class3.2.4 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_class3.2.4.csv", header = TRUE, sep = ",", stringsAsFactors
= FALSE)
242 method3.2.4 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/final_method3.2.4.csv", header = TRUE, sep = ",",
stringsAsFactors = FALSE)
243
244 package14_23 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/package14-23.csv", header = TRUE, sep = ",", stringsAsFactors =
FALSE)
245 class14_23 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/class14-23.csv", header = TRUE, sep = ",", stringsAsFactors =
FALSE)
246 method14_23 <- read.table("C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/method14-23.csv", header = TRUE, sep = ",", stringsAsFactors =
FALSE)
247
248 # combine all versions
249 package_all_version <-
rbind(package1.0,package1.1,package1.5,package1.6,package2.0.1,package2.0,pac
kage2.1,package2.2,package2.3.3,package2.3,package3.0,package3.1,package3.2.4
,package14_23)
```

```
250 class_all_version <-
rbind(class1.0,class1.1,class1.5,class1.6,class2.0.1,class2.0,class2.1,class2
.2,class2.3.3,class2.3,class3.0,class3.1,class3.2.4,class14_23)
251 method_all_version <-
rbind(method1.0,method1.1,method1.5,method1.6,method2.0.1,method2.0,method2.1
,method2.2,method2.3.3,method2.3,method3.0,method3.1,method3.2.4,method14_23)
252
253 # save final all version files
254 write.table(package_all_version, "C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/package_all_version.csv", sep = ",", row.names= FALSE)
255 write.table(class_all_version, "C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/class_all_version.csv", sep = ",", row.names= FALSE)
256 write.table(method_all_version, "C:/Users/mshen/Box Sync/Oracle-
Google/workstreams/22_API popularity analysis/android evolution/android
api/outcome/method_all_version.csv", sep = ",", row.names= FALSE)
257
258 method1_2 <- merge(x = method1.0,y = method1.1, by =
c("package_name","class_name","method_name"),all.x = TRUE)
259
260 setwd("C:/Users/mshen/Box Sync/Oracle-Google/workstreams/22_API
popularity analysis/android evolution/android api 1-23 clean")
261
262 method_all_version <- read.table("method_all_version.csv", header = TRUE,
sep = ",", stringsAsFactors = FALSE)
263
264 method4.0 <- subset(method_all_version,version == "4.0.1")
265 method4.2 <- subset(method_all_version,version == "4.2")
266 method4.4 <- subset(method_all_version,version == "4.4")
267 method5.0 <- subset(method_all_version,version == "5")
```

APPENDIX G – R Script for Calculating Package Changes across Java SE Versions

```
install.packages("dplyr")
library(dplyr)

m_unique <- java_stability[java_stability$count==1,]
m_multiple <- java_stability[java_stability$count > 1,]

#m_multiple$m_str_var <- paste(m_multiple$m_struct,m_multiple$m_var)

#write.csv(m_multiple,file='m_multiple.csv')

m_multiple_group <- group_by(m_multiple,class,package,method,version)
m_multiple_updated <- summarize(m_multiple_group,m_struct_c =
paste(m_struct, collapse = " "))

#android_stability$m_str_var <-
paste(android_stability$m_struct,android_stability$m_var)
#android_stability$m_char <- nchar(android_stability$m_str_var)


p <- c("java.applet",
       "java.awt",
       "java.awt.image",
       "java.awt.peer",
       "java.io",
       "java.lang",
       "java.net",
       "java.util",
       "java.awt.datatransfer",
       "java.awt.event",
       "java.beans",
       "java.lang.reflect",
       "java.math",
       "java.rmi",
       "java.rmi.dgc",
       "java.rmi.registry",
       "java.rmi.server",
       "java.security",
       "java.security.acl",
       "java.security.interfaces",
       "java.sql",
       "java.text",
       "java.text.resources",
       "java.util.zip",
       "java.awt.color",
       "java.awt.dnd",
       "java.awt.dnd.peer",
       "java.awt.font",
       "java.awt.geom",
       "java.awt.im",
       "java.awt.image.renderable",
       "java.awt.print",
       "java.beans.beancontext",
       "java.lang.ref",
       "java.rmi.activation",
       "java.security.cert",
```

```
        "java.security.spec",
        "java.util.jar",
        "javax.accessibility",
        "javax.swing",
        "javax.swing.border",
        "javax.swing.colorchooser",
        "javax.swing.event",
        "javax.swing.filechooser",
        "javax.swing.plaf",
        "javax.swing.plaf.basic",
        "javax.swing.plaf.metal",
        "javax.swing.plaf.multi",
        "javax.swing.table",
        "javax.swing.text",
        "javax.swing.text.html",
        "javax.swing.text.html.parser",
        "javax.swing.text.rtf",
        "javax.swing.tree",
        "javax.swing.undo",
        "org.omg.CORBA",
        "org.omg.CORBA.DynAnyPackage",
        "org.omg.CORBA.ORBPackage",
        "org.omg.CORBA.portable",
        "org.omg.CORBA.TypeCodePackage",
        "org.omg.CosNaming",
        "org.omg.CosNaming.NamingContextPackage",
        "java.awt.im.spi",
        "javax.naming",
        "javax.naming.directory",
        "javax.naming.event",
        "javax.naming.ldap",
        "javax.naming.spi",
        "org.omg.CORBA_2_3",
        "org.omg.CORBA_2_3.portable",
        "org.omg.SendingContext",
        "org.omg.stub.java.rmi",
        "java.nio",
        "java.nio.channels",
        "java.nio.channels.spi",
        "java.nio.charset",
        "java.nio.charset.spi",
        "java.util.logging",
        "java.util.prefs",
        "java.util.regex",
        "javax.imageio",
        "javax.imageio.event",
        "javax.imageio.metadata",
        "javax.imageio.plugins.jpeg",
        "javax.imageio.spi",
        "javax.imageio.stream",
        "javax.print",
        "javax.print.attribute",
        "javax.print.attribute.standard",
        "javax.print.event",
        "javax.rmi",
        "javax.rmi.CORBA",
        "javax.security.auth",
```

```
111         "javax.security.auth.callback",
112         "javax.security.auth.kerberos",
113         "javax.security.auth.login",
114         "javax.security.auth.spi",
115         "javax.security.auth.x500",
116         "javax.sql",
117         "javax.xml.parsers",
118         "javax.xml.transform",
119         "javax.xml.transform.dom",
120         "javax.xml.transform.sax",
121         "javax.xml.transform.stream",
122         "org.apache.crimson.jaxp",
123         "org.apache.crimson.parser",
124         "org.apache.crimson.tree",
125         "org.apache.crimson.util",
126         "org.apache.xalan.client",
127         "org.apache.xalan.extensions",
128         "org.apache.xalan.lib",
129         "org.apache.xalan.lib.sql",
130         "org.apache.xalan.processor",
131         "org.apache.xalan.res",
132         "org.apache.xalan.serialize",
133         "org.apache.xalan.templates",
134         "org.apache.xalan.trace",
135         "org.apache.xalan.transformer",
136         "org.apache.xalan.xslt",
137         "org.apache.xml.dtm",
138         "org.apache.xml.dtm.ref",
139         "org.apache.xml.dtm.ref.dom2dtm",
140         "org.apache.xml.dtm.ref.sax2dtm",
141         "org.apache.xml.utils",
142         "org.apache.xml.utils.res",
143         "org.apache.xml.utils.synthetic",
144         "org.apache.xml.utils.synthetic.reflection",
145         "org.apache.xpath",
146         "org.apache.xpath.axes",
147         "org.apache.xpath.compiler",
148         "org.apache.xpath.functions",
149         "org.apache.xpath.objects",
150         "org.apache.xpath.operations",
151         "org.apache.xpath.patterns",
152         "org.apache.xpath.res",
153         "org.ietf.jgss",
154         "org.omg.CosNaming.NamingContextExtPackage",
155         "org.omg.Dynamic",
156         "org.omg.DynamicAny",
157         "org.omg.DynamicAny.DynAnyFactoryPackage",
158         "org.omg.DynamicAny.DynAnyPackage",
159         "org.omg.IOP",
160         "org.omg.IOP.CodecFactoryPackage",
161         "org.omg.IOP.CodecPackage",
162         "org.omg.Messaging",
163         "org.omg.PortableInterceptor",
164         "org.omg.PortableInterceptor.ORBInitInfoPackage",
165         "org.omg.PortableServer",
166         "org.omg.PortableServer.CurrentPackage",
167         "org.omg.PortableServer.POAManagerPackage",
```

```
168         "org.omg.PortableServer.POAPackage",
169         "org.omg.PortableServer.portable",
170         "org.omg.PortableServer.ServantLocatorPackage",
171         "org.w3c.dom",
172         "org.w3c.dom.css",
173         "org.w3c.dom.events",
174         "org.w3c.dom.html",
175         "org.w3c.dom.stylesheets",
176         "org.w3c.dom.traversal",
177         "org.w3c.dom.views",
178         "org.xml.sax",
179         "org.xml.sax.ext",
180         "org.xml.sax.helpers",
181         "java.lang.annotation",
182         "java.lang.instrument",
183         "java.lang.management",
184         "java.util.concurrent",
185         "java.util.concurrent.atomic",
186         "java.util.concurrent.locks",
187         "javax.imageio.plugins.bmp",
188         "javax.management",
189         "javax.management.loading",
190         "javax.management.modelmbean",
191         "javax.management.monitor",
192         "javax.management.openmbean",
193         "javax.management.relation",
194         "javax.management.remote",
195         "javax.management.remote.rmi",
196         "javax.management.timer",
197         "javax.rmi.ssl",
198         "javax.security.sasl",
199         "javax.sound.midi",
200         "javax.sound.midi.spi",
201         "javax.sound.sampled",
202         "javax.sound.sampled.spi",
203         "javax.sql.rowset",
204         "javax.sql.rowset.serial",
205         "javax.sql.rowset.spi",
206         "javax.swing.plaf.synth",
207         "javax.xml",
208         "javax.xml.datatype",
209         "javax.xml.namespace",
210         "javax.xml.validation",
211         "javax.xml.xpath",
212         "org.w3c.dom.bootstrap",
213         "org.w3c.dom.ls",
214         "org.w3c.dom.ranges",
215         "java.text.spi",
216         "java.util.spi",
217         "javax.annotation",
218         "javax.annotation.processing",
219         "javax.lang.model",
220         "javax.lang.model.element",
221         "javax.lang.model.type",
222         "javax.lang.model.util",
223         "programelements",
224         "types",
```

```
225         "javax.script",
226         "javax.tools",
227         "javax.xml.bind",
228         "javax.xml.bind.annotation",
229         "javax.xml.bind.annotation.adapters",
230         "javax.xml.bind.attachment",
231         "javax.xml.bind.helpers",
232         "javax.xml.bind.util",
233         "javax.xml.crypto",
234         "javax.xml.crypto.dom",
235         "javax.xml.crypto.dsig",
236         "javax.xml.crypto.dsig.dom",
237         "javax.xml.crypto.dsig.keyinfo",
238         "javax.xml.crypto.dsig.spec",
239         "javax.xml.soap",
240         "javax.xml.stream",
241         "javax.xml.stream.events",
242         "javax.xml.stream.util",
243         "javax.xml.transform.stax",
244         "javax.xml.ws",
245         "javax.xml.ws.handler",
246         "javax.xml.ws.handler.soap",
247         "javax.xml.ws.http",
248         "javax.xml.ws.soap",
249         "javax.xml.ws.spi",
250         "org.w3c.dom.xpath",
251         "java.lang.invoke",
252         "java.nio.file",
253         "java.nio.file.attribute",
254         "java.nio.file.spi",
255         "javax.security.cert",
256         "javax.swing.plaf.nimbus",
257         "javax.xml.ws.spi.http",
258         "javax.xml.ws.wsaddressing",
259         "java.time",
260         "java.time.chrono",
261         "java.time.format",
262         "java.time.temporal",
263         "java.time.zone",
264         "java.util.function",
265         "java.util.stream")
266
267 # method change analysis
268
269 change_unique <- data.frame(matrix(ncol = 8, nrow = 248))
270 change_multiple <- data.frame(matrix(ncol = 8, nrow = 248))
271 change <- data.frame(matrix(ncol = 8, nrow = 248))
272
273 for (j in 1:248)
274 {
275   p_unique <- m_unique[m_unique$package == p[j],]
276   p_multiple <- m_multiple_updated[m_multiple_updated$package == p[j],]
277
278   for (i in 1:8)
279   {
280     p_change_unique <- merge(x = p_unique[p_unique$version==i,],y =
p_unique[p_unique$version==i+1,], by = c("class","method"),all = FALSE)
```

```
281     p_change_unique$change <- p_change_unique$m_struct.x ==
p_change_unique$m_struct.y
282     change_unique[j,i] <- nrow(p_change_unique[p_change_unique$change ==
FALSE,])
283
284     p_change_multiple <- merge(x = p_multiple[p_multiple$version==i,],y =
p_multiple[p_multiple$version==i+1,], by = c("class","method"),all = FALSE)
285     p_change_multiple$change <- p_change_multiple$m_struct_c.x ==
p_change_multiple$m_struct_c.y
286     change_multiple[j,i] <-
nrow(p_change_multiple[p_change_multiple$change == FALSE,])
287
288     change <- change_unique + change_multiple
289
290   }
291
292 }
293
294 write.csv(change,file='change.csv')
295
296
297 #method added and removed
298
299 java_maturity <- distinct(select(java_stability,package, class, method,
version))
300
301 method_add <- data.frame(matrix(ncol = 8, nrow = 248))
302 method_remove <- data.frame(matrix(ncol = 8, nrow = 248))
303
304 for (j in 1:248)
305 {
306   p_analysis <- java_maturity[java_maturity$package == p[j],]
307
308   for (i in 1:8)
309   {
310     p_merge <- merge(x = p_analysis[p_analysis$version==i,],y =
p_analysis[p_analysis$version==i+1,], by = c("package","class","method"),all
= TRUE)
311     method_add[j,i] <- nrow(p_merge[is.na(p_merge$version.x),])
312     method_remove[j,i] <- nrow(p_merge[is.na(p_merge$version.y),])
313
314   }
315
316 }
317
318 write.csv(method_add,file='method_add.csv')
319 write.csv(method_remove,file='method_remove.csv')
320
321 #write.csv(android_maturity,file = 'android_maturity.csv')
322
323 # method size for all API levels
324
325 method_size <- data.frame(matrix(ncol = 9, nrow = 248))
326
327 for (j in 1:248)
328 {
329   p_size <- java_maturity[java_maturity$package == p[j],]
```

```
330
331   for (i in 1:9)
332   {
333     method_size[j,i] <- nrow(p_size[p_size$version==i,])
334
335   }
336
337 }
338
339 write.csv(method_size,file='method_size.csv')
340
341 # 37 API size over versions
342
343 p_copied <- c("javax.sql",
344               "java.beans",
345               "java.lang.ref",
346               "java.net",
347               "java.util.logging",
348               "java.util",
349               "java.io",
350               "java.lang",
351               "java.lang.annotation",
352               "java.nio",
353               "java.nio.channels",
354               "java.nio.channels.spi",
355               "java.nio.charset",
356               "java.security",
357               "java.security.acl",
358               "java.security.cert",
359               "java.security.interfaces",
360               "java.sql",
361               "java.text",
362               "java.util.jar",
363               "java.util.prefs",
364               "java.util.zip",
365               "javax.crypto",
366               "javax.crypto.interfaces",
367               "javax.crypto.spec",
368               "javax.net",
369               "javax.security.auth",
370               "javax.security.auth.callback",
371               "javax.security.auth.login",
372               "javax.security.cert",
373               "java.nio.charset.spi",
374               "java.security.spec",
375               "java.util.regex",
376               "javax.net.ssl",
377               "javax.security.auth.x500",
378               "java.lang.reflect",
379               "java.awt.font")
```

APPENDIX H – R Script for Calculating Package Changes across Android Versions

```
install.packages("dplyr")
library(dplyr)

#pull in the data files
setwd("C:/Users/mshen/Box Sync/Oracle-Google/workstreams/34_API stability analysis/android stability")

android_stability <- read.csv("android stability input.csv", as.is = TRUE)

#group the unique methods in each version and get the unique method list
android_stability <- android_stability %>% group_by(class,package,method,version) %>% mutate(length(m_struct))
names(android_stability)[8] <- "count"

#get the methods with only 1 method strucure in each version
m_unique <- android_stability[android_stability$count==1,]

#get the method with more than 1 method structure in each version
m_multiple <- android_stability[android_stability$count > 1,]
m_multiple$m_str_var <- paste(m_multiple$m_struct,m_multiple$m_var)

write.csv(m_multiple,file='m_multiple.csv')

m_multiple_group <- group_by(m_multiple,class,package,method,version)
m_multiple_updated <- summarize(m_multiple_group,m_struct_c = paste(m_struct, collapse = " "))


#android_stability$m_str_var <- paste(android_stability$m_struct,android_stability$m_var)
#android_stability$m_char <- nchar(android_stability$m_str_var)


p <- c("android",
       "android.annotation",
       "android.content.res",
       "java.nio.channels",
       "java.nio.channels.spi",
       "android.database",
       "java.nio.charset",
       "java.nio.charset.spi",
       "java.security",
       "java.security.acl",
       "java.security.cert",
       "java.security.spec",
       "java.sql",
       "java.text",
       "java.util",
       "android.database.sqlite",
       "java.util.concurrent",
       "java.util.concurrent.atomic",
       "java.util.concurrent.locks",
       "java.util.jar",
       "java.util.logging",
```

```
        "java.util.prefs",
        "java.util.regex",
        "java.util.zip",
        "javax.crypto",
        "javax.crypto.spec",
        "android.graphics",
        "javax.microedition.khronos.egl",
        "javax.net",
        "javax.net.ssl",
        "javax.security.auth",
        "javax.security.auth.callback",
        "javax.security.auth.login",
        "javax.security.auth.x500",
        "javax.security.cert",
        "javax.sql",
        "javax.xml",
        "javax.xml.parsers",
        "junit.framework",
        "junit.runner",
        "org.apache.http",
        "org.apache.http.auth",
        "org.apache.http.auth.params",
        "org.apache.http.client",
        "org.apache.http.client.entity",
        "org.apache.http.client.methods",
        "org.apache.http.client.params",
        "org.apache.http.client.protocol",
        "org.apache.http.client.utils",
        "org.apache.http.conn",
        "org.apache.http.conn.params",
        "org.apache.http.conn.routing",
        "org.apache.http.conn.scheme",
        "org.apache.http.conn.ssl",
        "org.apache.http.conn.util",
        "org.apache.http.cookie",
        "org.apache.http.cookie.params",
        "org.apache.http.entity",
        "org.apache.http.impl",
        "org.apache.http.impl.auth",
        "org.apache.http.impl.client",
        "org.apache.http.impl.conn",
        "org.apache.http.impl.conn.tsccm",
        "org.apache.http.impl.cookie",
        "org.apache.http.impl.entity",
        "org.apache.http.impl.io",
        "org.apache.http.message",
        "org.apache.http.params",
        "android.app",
        "org.apache.http.protocol",
        "org.apache.http.util",
        "org.json",
        "org.w3c.dom",
        "org.xml.sax",
        "org.xml.sax.ext",
        "org.xml.sax.helpers",
        "org.xmlpull.v1",
        "org.xmlpull.v1.sax2",
```

```
108         "android.graphics.drawable",
109         "android.graphics.drawable.shapes",
110         "android.hardware",
111         "android.location",
112         "android.media",
113         "android.net",
114         "android.net.http",
115         "android.net.wifi",
116         "android.opengl",
117         "android.os",
118         "android.preference",
119         "android.provider",
120         "android.sax",
121         "android.telephony",
122         "android.telephony.gsm",
123         "android.test",
124         "android.test.mock",
125         "android.test.suitebuilder",
126         "android.test.suitebuilder.annotation",
127         "android.text",
128         "android.text.method",
129         "android.text.style",
130         "android.content",
131         "android.text.util",
132         "android.util",
133         "android.view",
134         "android.view.animation",
135         "android.webkit",
136         "android.widget",
137         "dalvik.annotation",
138         "dalvik.system",
139         "java.awt.font",
140         "java.io",
141         "android.content.pm",
142         "java.lang",
143         "java.lang.annotation",
144         "java.lang.ref",
145         "java.lang.reflect",
146         "java.math",
147         "java.net",
148         "java.nio",
149         "com.android.internal.util",
150         "dalvik.bytecode",
151         "java.security.interfaces",
152         "javax.crypto.interfaces",
153         "javax.microedition.khronos.opengles",
154         "org.apache.commons.logging",
155         "org.apache.http.io",
156         "android.inputmethodservice",
157         "android.speech",
158         "android.text.format",
159         "android.appwidget",
160         "android.view.inputmethod",
161         "java.beans",
162         "android.gesture",
163         "android.accessibilityservice",
164         "android.speech.tts",
```

```
        "android.view.accessibility",
        "android.accounts",
        "android.telephony.cdma",
        "android.bluetooth",
        "android.service.wallpaper",
        "javax.xml.datatype",
        "javax.xml.namespace",
        "javax.xml.transform",
        "javax.xml.transform.dom",
        "javax.xml.transform.sax",
        "javax.xml.transform.stream",
        "javax.xml.validation",
        "javax.xml.xpath",
        "org.w3c.dom.ls",
        "android.app.admin",
        "android.app.backup",
        "android.media.audiofx",
        "android.net.sip",
        "android.nfc",
        "android.nfc.tech",
        "android.os.storage",
        "android.drm",
        "android.animation",
        "android.renderscript",
        "android.hardware.usb",
        "android.mtp",
        "android.net.rtp",
        "android.media.effect",
        "android.net.wifi.p2p",
        "android.security",
        "android.service.textservice",
        "android.view.textservice",
        "android.hardware.input",
        "android.net.nsd",
        "android.net.wifi.p2p.nsd",
        "android.hardware.display",
        "android.service.dreams",
        "android.hardware.location",
        "android.service.notification",
        "android.graphics.pdf",
        "android.nfc.cardemulation",
        "android.print",
        "android.print.pdf",
        "android.printservice",
        "android.transition",
        "android.app.job",
        "android.app.usage",
        "android.bluetooth.le",
        "android.hardware.camera2",
        "android.hardware.camera2.params",
        "android.media.browse",
        "android.media.projection",
        "android.media.session",
        "android.media.tv",
        "android.service.media",
        "android.service.restrictions",
        "android.service.voice",
```

```
222         "android.system",
223         "android.telecom",
224         "android.service.carrier",
225         "android.app.assist",
226         "android.hardware.fingerprint",
227         "android.media.midi",
228         "android.security.keystore",
229         "android.service.chooser")
230
231 # method change analysis
232
233 change_unique <- data.frame(matrix(ncol = 22, nrow = 200))
234 change_multiple <- data.frame(matrix(ncol = 22, nrow = 200))
235 change <- data.frame(matrix(ncol = 22, nrow = 200))
236
237 for (j in 1:200)
238 {
239   p_unique <- m_unique[m_unique$package == p[j],]
240   p_multiple <- m_multiple_updated[m_multiple_updated$package == p[j],]
241
242   for (i in 1:22)
243   {
244       p_change_unique <- merge(x = p_unique[p_unique$version==i,],y =
p_unique[p_unique$version==i+1,], by = c("class","method"),all = FALSE)
245       p_change_unique$change <- p_change_unique$m_struct.x ==
p_change_unique$m_struct.y
246       change_unique[j,i] <- nrow(p_change_unique[p_change_unique$change
== FALSE,])
247
248       p_change_multiple <- merge(x = p_multiple[p_multiple$version==i,],y
= p_multiple[p_multiple$version==i+1,], by = c("class","method"),all = FALSE)
249       p_change_multiple$change <- p_change_multiple$m_struct_c.x ==
p_change_multiple$m_struct_c.y
250       change_multiple[j,i] <-
nrow(p_change_multiple[p_change_multiple$change == FALSE,])
251
252       change <- change_unique + change_multiple
253
254   }
255
256 }
257
258 write.csv(change,file='change.csv')
259
260 # method change for API level 13-14
261
262 android_1314 <- read.csv("13_14_input.csv", as.is = TRUE)
263 android_1314 <- android_1314 %>% group_by(class,package,method,version)
%>% mutate(length(m_struct))
264 names(android_1314)[7] <- "count"
265
266 m_unique_1314 <- android_1314[android_1314$count==1,]
267 m_multiple_1314 <- android_1314[android_1314$count > 1,]
268
269 m_multiple_1314group <-
group_by(m_multiple_1314,class,package,method,version)
```

```
270 m_multiple_1314updated <- summarize(m_multiple_1314group,m_struct_c =
paste(m_struct, collapse = " "))
271
272 change_unique1314 <- data.frame(matrix(ncol = 1, nrow = 200))
273 change_multiple1314 <- data.frame(matrix(ncol = 1, nrow = 200))
274 change1314 <- data.frame(matrix(ncol = 1, nrow = 200))
275
276 for (j in 1:200)
277 {
278   p_unique1314 <- m_unique_1314[m_unique_1314$package == p[j],]
279   p_multiple1314 <- m_multiple_1314updated[m_multiple_1314updated$package
== p[j],]
280
281   p_change_unique1314 <- merge(x =
p_unique1314[p_unique1314$version==13,],y =
p_unique1314[p_unique1314$version==14,], by = c("class","method"),all =
FALSE)
282   p_change_unique1314$change <- p_change_unique1314$m_struct.x ==
p_change_unique1314$m_struct.y
283   change_unique1314[j,1] <-
nrow(p_change_unique1314[p_change_unique1314$change == FALSE,])
284
285   p_change_multiple1314 <- merge(x =
p_multiple1314[p_multiple1314$version==13,],y =
p_multiple1314[p_multiple1314$version==14,], by = c("class","method"),all =
FALSE)
286   p_change_multiple1314$change <- p_change_multiple1314$m_struct_c.x ==
p_change_multiple1314$m_struct_c.y
287   change_multiple1314[j,1] <-
nrow(p_change_multiple1314[p_change_multiple1314$change == FALSE,])
288
289   change1314 <- change_unique1314 + change_multiple1314
290
291 }
292
293 write.csv(change1314,file='change13_14.csv')
294
295
296 #method added and removed
297
298 android_maturity <- distinct(select(android_stability,package, class,
method, version))
299
300 method_add <- data.frame(matrix(ncol = 22, nrow = 200))
301 method_remove <- data.frame(matrix(ncol = 22, nrow = 200))
302
303 for (j in 1:200)
304 {
305   p_analysis <- android_maturity[android_maturity$package == p[j],]
306
307   for (i in 1:22)
308   {
309     p_merge <- merge(x = p_analysis[p_analysis$version==i,],y =
p_analysis[p_analysis$version==i+1,], by = c("package","class","method"),all
= TRUE)
310     method_add[j,i] <- nrow(p_merge[is.na(p_merge$version.x),])
311     method_remove[j,i] <- nrow(p_merge[is.na(p_merge$version.y),])
```

```
312
313   }
314
315 }
316
317 write.csv(method_add,file='method_add.csv')
318 write.csv(method_remove,file='method_remove.csv')
319
320 write.csv(android_maturity,file = 'android_maturity.csv')
321
322 # method size for all API levels
323
324 method_size <- data.frame(matrix(ncol = 23, nrow = 200))
325
326 for (j in 1:200)
327 {
328   p_size <- android_maturity[android_maturity$package == p[j],]
329
330   for (i in 1:23)
331   {
332    method_size[j,i] <- nrow(p_size[p_size$version==i,])
333
334   }
335
336 }
337
338 write.csv(method_size,file='method_size.csv')
339
340 # get the list of changed methods over all versions in the coplies APIs
341
342 setwd("C:/Users/mshen/Box Sync/Oracle-Google/workstreams/34_API stability
analysis/android stability")
343
344 android_stability <- read.csv("android stability input.csv", as.is =
TRUE)
345 android_stability <- android_stability %>%
group_by(class,package,method,version) %>% mutate(length(m_struct))
346 names(android_stability)[8] <- "count"
347
348 m_unique <- android_stability[android_stability$count==1,]
349 m_multiple <- android_stability[android_stability$count > 1,]
350
351 m_multiple_group <- group_by(m_multiple,class,package,method,version)
352 m_multiple_updated <- summarize(m_multiple_group,m_struct_c =
paste(m_struct, collapse = " "))
353
354 p_copied <- c("javax.sql",
355               "java.beans",
356               "java.lang.ref",
357               "java.net",
358               "java.util.logging",
359               "java.util",
360               "java.io",
361               "java.lang",
362               "java.lang.annotation",
363               "java.nio",
364               "java.nio.channels",
```

```
365                "java.nio.channels.spi",
366                "java.nio.charset",
367                "java.security",
368                "java.security.acl",
369                "java.security.cert",
370                "java.security.interfaces",
371                "java.sql",
372                "java.text",
373                "java.util.jar",
374                "java.util.prefs",
375                "java.util.zip",
376                "javax.crypto",
377                "javax.crypto.interfaces",
378                "javax.crypto.spec",
379                "javax.net",
380                "javax.security.auth",
381                "javax.security.auth.callback",
382                "javax.security.auth.login",
383                "javax.security.cert",
384                "java.nio.charset.spi",
385                "java.security.spec",
386                "java.util.regex",
387                "javax.net.ssl",
388                "javax.security.auth.x500",
389                "java.lang.reflect",
390                "java.awt.font")
391
392 # get the method change list between each API level
393
394 p_change_1 <- data.frame(class = character(),method =
character(),c_struct_pre = character(),
395                          m_struct_pre = character(),m_var_pre =
character(),version_pre = numeric(),
396                          package_pre =
character(),count_pre=numeric(),c_struct_post = character(),
397                          m_struct_post = character(),m_var_post =
character(),version_post = numeric(),
398                          package_post = character(),count_post =
numeric(),change = logical())
399 p_change_2 <- data.frame(class = character(),method =
character(),package_pre = character(),version_pre = numeric(),
400                          m_struct_pre = character(),package_post =
character(),version_post = numeric(),
401                          m_struct_post = character(),change = logical())
402
403 for (j in 1:37)
404 {
405   p_unique <- m_unique[m_unique$package ==p_copied[j],]
406   p_multiple <- m_multiple_updated[m_multiple_updated$package ==
p_copied[j],]
407
408   for (i in 1:22)
409   {
410     p_change_unique <- merge(x = p_unique[p_unique$version==i,],y =
p_unique[p_unique$version==i+1,], by = c("class","method"),all = FALSE)
411     p_change_unique$change <- p_change_unique$m_struct.x ==
p_change_unique$m_struct.y
```

```
412     p_change_unique <- p_change_unique[p_change_unique$change == FALSE,]
413     p_change_1 <- rbind(p_change_1,p_change_unique)
414
415     p_change_multiple <- merge(x = p_multiple[p_multiple$version==i,],y =
p_multiple[p_multiple$version==i+1,], by = c("class","method"),all = FALSE)
416     p_change_multiple$change <- p_change_multiple$m_struct_c.x ==
p_change_multiple$m_struct_c.y
417     p_change_multiple <- p_change_multiple[p_change_multiple$change ==
FALSE,]
418     p_change_2 <- rbind(p_change_2,p_change_multiple)
419
420   }
421
422 }
423
424 write.csv(p_change_1,file='p_change_1.csv')
425 write.csv(p_change_2,file='p_change_2.csv')
426
427 # get the list of change from API level 13 to 14
428
429 android_1314 <- read.csv("13_14_input.csv", as.is = TRUE)
430 android_1314 <- android_1314 %>% group_by(class,package,method,version)
%>% mutate(length(m_struct))
431 names(android_1314)[7] <- "count"
432
433 m_unique_1314 <- android_1314[android_1314$count==1,]
434 m_multiple_1314 <- android_1314[android_1314$count > 1,]
435
436 m_multiple_1314group <-
group_by(m_multiple_1314,class,package,method,version)
437 m_multiple_1314updated <- summarize(m_multiple_1314group,m_struct_c =
paste(m_struct, collapse = " "))
438
439 p_change_1_1314 <- data.frame(class = character(),method =
character(),c_struct_pre = character(),
440                               m_struct_pre = character(),m_var_pre =
character(),version_pre = numeric(),
441                               package_pre =
character(),count_pre=numeric(),c_struct_post = character(),
442                               m_struct_post = character(),m_var_post =
character(),version_post = numeric(),
443                               package_post = character(),count_post =
numeric(),change = logical())
444 p_change_2_1314 <- data.frame(class = character(),method =
character(),package_pre = character(),version_pre = numeric(),
445                               m_struct_pre = character(),package_post =
character(),version_post = numeric(),
446                               m_struct_post = character(),change = logical())
447
448
449 for (j in 1:37)
450 {
451   p_unique1314 <- m_unique_1314[m_unique_1314$package == p_copied[j],]
452   p_multiple1314 <- m_multiple_1314updated[m_multiple_1314updated$package
== p_copied[j],]
453
```

```
454   p_change_unique1314 <- merge(x =
p_unique1314[p_unique1314$version==13,],y =
p_unique1314[p_unique1314$version==14,], by = c("class","method"),all =
FALSE)
455   p_change_unique1314$change <- p_change_unique1314$m_struct.x ==
p_change_unique1314$m_struct.y
456   p_change_unique1314 <- p_change_unique1314[p_change_unique1314$change
== FALSE,]
457   p_change_1_1314 <- rbind(p_change_1_1314,p_change_unique1314)
458
459   p_change_multiple1314 <- merge(x =
p_multiple1314[p_multiple1314$version==13,],y =
p_multiple1314[p_multiple1314$version==14,], by = c("class","method"),all =
FALSE)
460   p_change_multiple1314$change <- p_change_multiple1314$m_struct_c.x ==
p_change_multiple1314$m_struct_c.y
461   p_change_multiple1314 <-
p_change_multiple1314[p_change_multiple1314$change == FALSE,]
462   p_change_2_1314 <- rbind(p_change_2_1314,p_change_multiple1314)
463
464 }
465
466 write.csv(p_change_1_1314,file='p_change_1_1314.csv')
467 write.csv(p_change_2_1314,file='p_change_2_1314.csv')
```

PROOF OF SERVICE BY KITEWORKS

I, José E. Valdés, am over the age of eighteen years old and not a party to the within-entitled action. My place of employment and business address is Orrick, Herrington & Sutcliffe LLP, 1000 Marsh Road, Menlo Park, California 94025.

On February 8, 2016, I served the following documents:

EXPERT REPORT OF CHRIS F. KEMERER, Ph.D. REGARDING FAIR USE AND REBUTTAL TO GOOGLE'S OPENING EXPERT REPORTS

on the interested parties in this action by electronic service [Fed. Rule Civ. Proc. 5(b)] by electronically mailing a true and correct copy, pursuant to the parties agreement, to the following email addresses:

DALVIK-KVN@kvn.com
JCooper@fbm.com
gglas@fbm.com

I declare under penalty of perjury under the laws of the State of California and the United States that the foregoing is true and correct.

Executed on February 8, 2016, at San Francisco, California.

/s/ José E. Valdés
José E. Valdés